ELSEVIER

European Journal of Operational Research 100 (1997) 180–191

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

A tabu search algorithm for the multi-trip vehicle routing and scheduling problem

José Brandão [a,*], Alan Mercer [b]

[a] *Gestão e Administração Pública, EEG, Universidade do Minho, 4709 Braga codex, Portugal*
[b] *Department of Management Science, Lancaster University, Lancaster, LA1 4YX, UK*

Received 4 May 1996; accepted 7 November 1996

Abstract

This paper describes a novel tabu search heuristic for the multi-trip vehicle routing and scheduling problem (MTVRSP). The method was developed to tackle real distribution problems, taking into account most of the constraints that appear in practice. In the MTVRSP, besides the constraints that are common to the basic vehicle routing problem, the following ones are present: during each day a vehicle can make more than one trip; the customers impose delivery time windows; the vehicles have different capacities considered in terms of both volume and weight; the access to some customers is restricted to some vehicles; the drivers' schedules must respect the maximum legal driving time per day and the legal time breaks; the unloading times are considered. © 1997 Elsevier Science B.V.

Keywords: Vehicle routing; Vehicle scheduling; Tabu search; Heuristics

1. Introduction

In the distribution of goods or services, there exists a problem that is known as the *vehicle routing problem*. In fact there is a wide variety of situations and therefore the problem is not unique but a vast class of problems, each one with its own characteristics and constraints. Most of the time the *scheduling* is also a constraint of these problems, so that the name which seems more appropriate for such a class of real problems is the *vehicle routing and scheduling problem* (VRSP). The extensive literature dedicated to the VRSP is mainly due to its many applications and economic importance, and to the difficulty of solving it. In fact the VRSP is classified as a *NP-hard* problem (Lenstra and Rinnooy Kan, 1981; Savelsbergh, 1988).

The VRSP studied here is defined through the *basic vehicle routing problem* (VRP), where each of a given number of vehicles with the same capacity makes exactly one trip per day. Each trip consists of departing from the depot, visiting one or more customers, and returning to the depot. Each customer requiring goods is visited once, and only once, by one of the vehicles. The multi-trip VRSP (MTVRSP) is similar to the VRP, but has a larger number of constraints: during each day a vehicle can make more than one trip; the customers impose delivery time windows; the vehicles have different capacities (by volume and by weight); vehicles can be hired if the company has insufficient on any day; the access to some customers is restricted to some vehicles; the

* Corresponding author.

PII S0377-2217(97)00010-6

drivers' schedules must respect the maximum legal driving time per day and the legal time breaks; the unloading times are taken into account. The realistic character of the method proposed here is enhanced by the fact that real distances are used, based on a road network, created with the digital maps of the Ordnance Survey. The travelling times are also calculated through this network, assuming an average speed for each type of road and for each type of vehicle. Besides, the objective function considers all the transportation costs: fuel consumption, maintenance costs, drivers' wages, fixed costs of hired vehicles. A very important feature of the proposed algorithm is that it takes into account one constraint that is very common in practice and scarcely studied, namely the possibility of a vehicle making several trips per day. This constraint can be of enormous economic importance to some companies. The costs that are implicit in the objective function and some details of the algorithm are omitted due to space limitations but can be found in Brandão (1994).

Taillard et al. (1996) have very recently described the following heuristic for dealing with the multiple use of vehicles in the VRP. A large set of good vehicle routes satisfying the VRP constraints is generated. Then a subset of these routes is selected using an enumerative algorithm. Finally, the selected routes are assembled into feasible working days using several applications of a bin packing algorithm. A very different approach is required for the practical problem which this paper addresses, not least because of the need to satisfy time windows and other constraints with vehicles of different sizes.

This paper describes an algorithm developed for the MTVRSP, using tabu search (TSAMTVRSP). The motivation was threefold: to be capable of solving real distribution problems with practical constraints and actual costs; to obtain high quality solutions; and to require small computing times because the distribution plans have to be produced daily.

Tabu search (TS) was proposed and developed by Glover (see Glover, 1989, 1990), Glover et al. (1993), inspired by the principles of artificial intelligence. One of those principles is the use of the memory for the resolution of difficult problems, and this is the most important characteristic of the TS method. The main concepts used in TS are: attributes, tabu list, tabu list size, aspiration criteria, intensification, diversification, target analysis, strategic oscillation, neighbourhood (of one solution) and neighbourhood size, moves and evaluation of the moves. TS has been applied intensively for various types of optimisation problems, with very good results. Nevertheless TS also has some weaknesses and many aspects are ill-defined. The application of TS has to be tailored to the type of problem and even to its size. A natural consequence of being a recent technique is that the practical application of these concepts depends very much on intuition and experimentation with concrete problems. The theoretical evolution of TS in terms of defining more precisely its domain, the practical meaning of its concepts and the choice of the parameters will need a lot of experimentation. Therefore, the present paper is also intended to make a small contribution in this respect.

The TSAMTVRSP algorithm takes advantage of both TS theory and an algorithm developed by Gendreau et al. (1992) for the travelling salesman problem (TSP) called GENI, which quickly gives very good solutions. Its behaviour in terms of solution quality and computing time, depends very much on a parameter (p) which defines the search neighbourhood. According to the authors, the best compromise between speed and quality seems to be $p = 5$. As each route of the MTVRSP is a TSP with time windows (TSPW), so GENI has been adapted by the authors for the TSPW but the same name has been retained. Gendreau et al. (1996) also produced an algorithm for the TSPW that has many of the same features. However, it cannot be applied directly in this research because infeasible solutions are also allowed in phase 2 of the TSAMTVRSP algorithm.

The TSAMTVRSP algorithm consists of three phases, which are applied sequentially, with the initial solution of each phase being the best solution found in the previous phase. The initial solution for phase 1 comes from a different kind of algorithm which takes into account all the routing but not the scheduling constraints. This structure drives the initial solution progressively towards the best feasible solution space. Each route is treated separately in phase 1, trying to eliminate the time window infeasibilities within individual routes and maintaining the travelling time as short as possible. Phase 2 is intended to find a feasible solution with the least travelling time but feasible and infeasible solutions

can be explored during the search process. Only feasible regions of the solution space are searched in phase 3 and the objective is to find a solution with the least variable cost.

2. Notation and formulae

V is a set of vertices (the calls plus the depot), $V = \{0,1,\ldots,N\}$, where 0 is the depot and N is the total number of calls to be made. A is the set of arcs connecting all the V vertices (or nodes).

u_i time unloading at call i, $i \in V$.
t_{ij} travel time between i and j, plus u_i, $(i,j) \in A$.
e_i earliest start time of the delivery at call i, $i \in V$.
l_i latest finish time of the delivery at call i, $i \in V$.
a_i actual arrival time of the vehicle at call i, $i \in V$.
d_i departure time of the vehicle from call i, $i \in V$.

Route (or trip): set constituted by the depot and by one or more calls, in a given sequence, which is called the sequence of routing or travelling. The route always begins and ends at the same depot. Each route r is represented as $r = [0,1,\ldots,n,n+1]$, where the origin, 0, and the destination, $n+1$, are always the depot, and the numbers, $1,\ldots,n$, represent the calls (or orders), which are visited in this sequence.

Tour: set of routes assigned to the same vehicle (driver).

Nearest: this adjective may be interpreted either as a distance, or a time, or a variable cost. If nothing is stated to the contrary, it is interpreted as a distance. In the algorithms, sometimes it is more convenient to use order than call. When this happens, if it is stated that an order is nearest to another, this really means that the respective calls are nearest to each other.

Nr total number of routes in the solution.
Nt total number of tours in the solution.
W_i waiting time at call i, $i \in V$. This is the time which the driver must wait if he arrives at the call before the earliest start of the delivery.
Sf_i forward slack of the order i, $i \in V$. This variable measures for each call i, in the current route, the amount of time that the arrival of the vehicle at i can be delayed and still comply with the delivery time window at i.

h number of routes in the tour to which the route r (the one that is being analysed) belongs.
L_i lateness of the order i, $i \in V$.
L total lateness in a route. If it is necessary to distinguish between routes, it is written $L(i)$ for route i.
ni number of infeasible orders in a route.
nf number of feasible orders in a route. An order in a route is said to be infeasible relative to the delivery time window if it is not delivered within this time window; otherwise it is feasible. This definition can be extended to the depot, because usually the depot is not open for some services, like loading the vehicles, during a period of the day.
T the net time of a driver's working day, i.e., the time that legally he can drive (actually under UK legislation, $T = 540$ min). Therefore, the legal obligatory time breaks during the normal working day are not included in T.
To legal overtime per working day.
B is the legal time break.
OB legal time break due to overtime.
D is a constant that penalises overtime (the value calculated and used is $D = 1$).

It is supposed that the delivery must be finished within the time window, but if the customer allows some tolerance, that window is enlarged accordingly.

The legal overtime (To), when expressed in min, is given by (1):

$$0 \le \mathrm{To} \le 60. \tag{1}$$

The waiting time at call i is given by Eq. (2). If the vehicle arrives late at the call, i.e., if $a_i > l_i$, it is supposed that $W_i = 0$:

$$W_i = \max\{0, e_i - a_i\}. \tag{2}$$

The time ($t(r)$) to travel route r (excluding the drivers' breaks) is given by the equivalent expres-

sions (3) and (4). It should be remembered that t_{ij} already includes the unloading time u_i, and that u_0 is the time of preparing the vehicle in the morning for the first route of the tour, and it is the time of loading the vehicle for the other routes:

$$t(r) = \sum_{i=0}^{n} [t_{ii+1} + W_i], \tag{3}$$

$$t(r) = a_{n+1} - a_0. \tag{4}$$

The remaining normal driving time (Tr_i) for the route i is given by Eq. (5). It is supposed that the driving time is allocated sequentially to the routes of the tour:

$$\mathrm{Tr}_i = T - \sum_{j=1}^{j<i}, \quad 1 \le i \le h. \tag{5}$$

Inequality (6) establishes the maximum travelling time of route i that keeps it feasible, where Tor_i is the remaining overtime for route i, which is given by (8):

$$t(i) \le \mathrm{Tr}_i + \mathrm{Tor}_i, \quad 1 \le i \le h. \tag{6}$$

The overtime (Ot_i or $\mathrm{Ot}(i)$) used by route i is given by Eq. (7). In a tour, the overtime is always allocated to the last routes of the tour. This means that if a route uses overtime, all the subsequent routes use overtime exclusively:

$$\mathrm{Ot}_i = \max\{0, t(i) - \mathrm{Tr}_i\}, \tag{7}$$

$$\mathrm{Tor}_i = \mathrm{To} - \sum_{j=1}^{j<i} \mathrm{Ot}_i, \quad 1 \le i \le h. \tag{8}$$

A tour is feasible relative to the maximum driving time if all the routes of the tour satisfy Eq. (6). For each route r, Eq. (9) to Eq. (14) hold:

$$d_i = a_i + W_i, \quad i = 0,1,\ldots,n+1, \tag{9}$$

$$a_{i+1} = d_i + t_{ii+1}, \quad i = 0,1,\ldots,n, \tag{10}$$

$$\mathrm{Sf}_i = l_i - a_i, \quad i = 0,1,\ldots,n+1, \tag{11}$$

$$e_i \le d_i \le l_i, \quad i = 0,1,\ldots,n+1, \tag{12}$$

$$L_i = \max\{0, d_i - l_i\}, \quad i = 0,1,\ldots,n+1. \tag{13}$$

If $L_i > 0$ then $\mathrm{Sf}_i < 0$ and Eq. (12) is not satisfied. In this case, order i is infeasible. The lateness for a route with ni infeasible orders is given by Eq. (14):

$$L = \sum_{i=1}^{\mathrm{ni}} d_i - l_i. \tag{14}$$

The overtime break (OB) defined in Eq. (15) is only assigned to the last route of each tour:

$$\mathrm{OB} = \begin{cases} 0, & \text{if} \quad \mathrm{Ot} = 0; \\ 45, & \text{if} \quad 0 < \mathrm{Ot} \le \mathrm{To}. \end{cases} \tag{15}$$

3. The structure of the TSAMTVRSP algorithm

An initial solution for the TSAMTVRSP is given by an algorithm described fully in Brandão (1994). This algorithm mixes two classical approaches: nearest neighbour and insertion. In a first stage, the routes are created in a sequential constructive manner, with the call introduced in the route being chosen based on its vicinity to all the calls already included. Besides, that call is selected from a set that contains only a given number of calls nearest to those already in the route. This constraint results in some calls remaining unrouted, so they are introduced in the second stage. In this stage, each of the unrouted orders is inserted in one of the existing routes. This is a parallel procedure, because all the routes compete for the same call, which is introduced into the one where the increase in the route length is minimised. The initial solution is feasible relative to all the routing constraints of the problem but it may be infeasible with regard to the scheduling constraints (time windows, maximum driving time, drivers' breaks). The legal breaks of a driver can be described in a simplified manner as follows: a driver cannot drive for more than 4.5 hours consecutively without having a break of 45 min. This break can be taken in a single period, two periods (one of 30 min and another of 15 min) or three periods of 15 min. Although, this constraint is conceptually easy to consider, it is very complicated to implement in the algorithm. A straightforward implementation will increase very much the computational complexity of the algorithms. The approach used to deal with the drivers' breaks, so as not to introduce any more computational complexity, is to *increase the travelling time*. The normal driving time is modified to $T' = T + B$ $(= 540 + 45)$, and all times of the *time*

matrix are multiplied by T'/T ($= 585/540$). After this, in order that the formulae do not need to be changed, T takes the value of T'. With this approach some slack is implicit in the schedule which may accommodate the real breaks that can be introduced in the final stage of the scheduling. Nevertheless, the real constraint is not equivalent and, in general, is tighter than that imposed by this method. One way of coping with this is to increase slightly the ratio (T'/T) without increasing T'. It should be noted that when using this approach, $t(r)$ is still calculated from expressions (3) or (4).

An alternative method of handling drivers' breaks was developed by Rochat and Semet (1994). The real constraints imposed by the Swiss work legislation were approximated by assuming a half hour break between 7.30am and 10.00am and a one hour break between 11.45am and 1.00pm. Then each break was considered to be a fictitious customer with service times of 30 and 60 min, respectively, during the corresponding time windows. This approach is stricter than both the Swiss and UK regulations require, but nevertheless it is valuable.

The principal objective of phase 1 is to try to make feasible each route that is infeasible because of the time window constraints, maintaining the travelling time as small as possible. In phase 2, two objectives are sought simultaneously: to make a solution feasible, in terms of time windows and maximum driving time; to reduce the solution cost as much as possible. The objective of phase 3 is exclusively to reduce the solution cost whilst maintaining the feasibility of the solution. The solution costs in phases 1 and 2 are measured and evaluated by the travelling time. In phase 3 the measure and evaluation of the solutions is more generic, with all the variable costs being included.

In the TSAMTVRSP, two kinds of moves are defined: the *insert* move and the *swap* move. The insert move consists of moving one variable of the candidate set from one position to another. The *swap* move consists of interchanging two variables of the candidate set. The combination of these two types of moves has been shown by Barnes and Laguna (1993) to be very efficient in scheduling problems.

The following subsections describe separately the characteristics of each phase of the TSAMTVRSP. However, the tabu list size and tabu attributes for the whole algorithm are described together in Section 3.4 because the three phases have many aspects in common.

3.1. Phase 1 of the TSAMTVRSP

Each route is considered separately in this phase and only the infeasible routes (i.e. the routes with one or more infeasible calls) are considered for improvement. The *set of candidates* (C) is composed of two disjoint subsets: an *infeasible* set (Ci) and a *feasible* set (Cf). Ci_s is a subset of Ci to be used in the swap moves. The cardinal of these sets for each route r is defined by Eq. (16) to Eq. (19). The members of Ci are the calls with lowest forward slack and the members of Cf are the calls with largest forward slack:

$$\#C = \left\lceil \frac{n}{c} \right\rceil, \tag{16}$$

$$\#\mathrm{Ci} = \min\{\mathrm{ni}, \#C\}, \tag{17}$$

$$\#\mathrm{Ci}_s = \min\left\{\mathrm{ni}, \frac{\#C}{2}\right\}, \tag{18}$$

$$\#\mathrm{Cf} = \min\{\mathrm{nf}, \#C - \#\mathrm{Ci}\}. \tag{19}$$

The value of c is determined by experience, with $c = 2$ being used in the present example.

Expression (20) defines the δ-neighbourhood of the calls belonging to the route. This means the neighbourhood of each call is defined by the δ nearest neighbours (vertices), in terms of distance. The term vertices is used to emphasise that the depot can also be a neighbour of any call, in the present context. The value of δ is defined by the constant c_1, which is again taken to be 2:

$$\delta = \left\lceil \frac{n}{c_1} \right\rceil. \tag{20}$$

The position of an order i in the route is designated by $\Phi(i)$. If $\Phi(i) < \Phi(j)$, this means that the order i is positioned earlier in the route than j and consequently it is visited earlier. The first position of any route is the position 0 that is occupied by the depot. The expression (21) defines the positional distance d of two calls in a route. If the calls are i and j, $d = |\Phi(i) - \Phi(j)|$. Again, c_2 has been taken to be 2:

$$d = \left\lceil \frac{n}{c_2} \right\rceil. \tag{21}$$

3.1.1. Trial moves

1. *Insert moves*

If $k \in \mathrm{Ci}$, a *trial* insert of k between $(i,j) \in r$ exists if the following two conditions are verified:

(a) $\Phi(j) = \Phi(i) + 1$, and $\Phi(j) < \Phi(k)$;

(b) i or j (or both) belong to the δ-neighbourhood of k.

The candidates for insert are all the members of Ci, as determined by Eq. (17).

2. *Swap moves*

The sets Ci_s and Cf are determined by Eq. (18) and Eq. (19), respectively. If $i \in \mathrm{Ci}_s$ and $j \in$ Cf, a *trial* swap of i and j exists if the two following conditions are both verified:

$\Phi(j) < \Phi(i)$;

$\Phi(i) - \Phi(j) \leq d$.

3.1.2. Evaluation of the moves and aspiration criteria

The evaluation function used in phase 1 (and phase 2 is similar) is an objective function that includes the travelling time and a penalty. This penalty is a measure of the infeasibility of the solution and therefore it acts as the force which drives the search in the feasible direction. The moves in phase 1 are evaluated by Eq. (22). This equation not only penalises the lateness but also the number of late calls.

The best solution found in phase 1 for each route r, which is the one passed on to phase 2, is defined by Eq. (23). The reason for using Eq. (23) is that Eq. (22) can distort the quality of the solutions, because of the coefficient $(\mathrm{ni})^{1/2}$:

$$F = t(r) + (\mathrm{ni})^{1/2} L(r) + D\mathrm{Ot}(r) + \mathrm{OB}(D+1), \tag{22}$$

$$F^* = t(r) + D\mathrm{Ot}(r) + \mathrm{OB}(D+1). \tag{23}$$

The *aspiration criteria* is the set of criteria which allows a move to be performed that is tabu in the current iteration. A tabu move can be overridden in phase 1 if it gives a value of F calculated by Eq. (22) that is less than the best known value of F.

3.2. Phase 2 of the TSAMTVRSP

3.2.1. Trial moves

The same two types of move also are made in phase 2 of the TSAMTVRSP: *insert* and *swap*. The set of candidates of each route is defined by Eq. (16), again with c taken to be 2 and the orders that constitute each set are chosen randomly. The set of all candidates is also designated by C.

1. *Insert moves*

The insert move consists of removing one order from one route (origin) and putting it into another route (destination). The destination route can be an existing route or a new route but the two routes must belong to different tours. The new route can be created in an existing tour, if this is admissible. If the new route is created in a new tour, it is assigned to a company vehicle if there is still one available; otherwise it is assigned to a vehicle, which may be hired for the day.

Creating new routes can be very important, because this may be the only way of making a solution feasible. This guarantees, if enough time is given to the search process, that a feasible solution is found.

A trial insert of an order $x_i \in r_i$ and $x_i \in C$, into r_j $(r_i \neq r_j)$ exists if and only if:

1. at least one order of r_j belongs to the δ-neighbourhood (δ nearest neighbours) of x_i, or r_j is a new route;
2. x_i can be visited by the vehicle that contains r_j;
3. the insertion of x_i in r_j does not cause an excess of load in the vehicle that contains r_j.

2. *Swap moves*

A swap move consists of exchanging two calls belonging to two different routes. A trial swap of an order $x_i \in r_i$ and $x_i \in \mathrm{C}$, with a $x_j \in r_j$ and $x_j \in C$ $(r_i \neq r_j)$ exists if and only if:

1. at least one order of r_j belongs to the δ-neighbourhood of x_i and at least one order of r_i belongs to the δ-neighbourhood of x_j;
2. x_i can be visited by the vehicle that contains x_j, and conversely;
3. the exchange of x_i and x_j does not cause an excess of load either in r_i or r_j;
4. at least one of the routes r_i or r_j contains more than one order;
5. r_i and r_j belong to two different tours.

The condition that r_i and r_j must belong to two different tours is only used to simplify the calculations.

The parameter δ is a way of limiting the likely number of potential moves without reducing the quality of the solutions. If δ is too high, it loses its meaning and purpose of diminishing the neighbourhood size. On the other hand, if δ is too small, it restricts the search unduly and it is less likely that a good solution will be found. A good trade-off may be to make δ equal to the parameter p of the GENI algorithm. Since a good choice for p is 5, $\delta = 5$ is also used. As in phase 1, the neighbourhood is constructed in terms of distance.

The rule (1) of the insert and swap moves may prevent some calls from leaving their route, if this route has more than δ calls, so impeding the search for better alternatives. In order to take this into account, a complementary procedure (*part 2*) to the previous procedure (*part 1*) is provided within phase 2.

Part 2. The equations below are used to calculate $F1$ and $F2$, which represent the cost of a feasible and an infeasible solution, respectively. The value of $F1$ is based on the travelling time, the overtime, the break time and a constant that takes into account if the vehicle is hired. $F2$ includes the same factors as $F1$ and also one to evaluate the infeasibility:

$$F1 = \sum_{i=1}^{Nr} \beta\{t(i) + DOt(i) + OB(D+1)\}, \quad (24)$$

$$F2 = \sum_{i=1}^{Nr} \beta\{t(i) + DOt(i) + OB(D+1)\} + PL(i), \quad (25)$$

$$\beta = \begin{cases} 1.5, & \text{if the route } i \text{ is assigned to a hired vehicle;} \\ 1, & \text{otherwise.} \end{cases} \quad (26)$$

The penalty P is initially equal to 1. If during α consecutive iterations, all the solutions have been infeasible, P is multiplied by 2. Conversely, if all the solutions have been feasible during the last α consecutive iterations, P is divided by 2. First $\alpha = 10$ is tested. Using P to create strategic oscillation has been applied successfully by various authors, such as Kelly et al. (1993), Mooney and Rardin (1993) and Gendreau et al. (1994), who vary P in a VRP algorithm in the same way as in the present algorithm. However, the current concern is not simply to diversify the search but rather to drive it towards the feasible region, which it does very successfully.

Part 1 is replaced by *part 2* when a given number of iterations (*limit*) is executed without improving either $F1^*$ or $F2^*$ ($F1^*$ is the best known value of $F1$ from Eq. (24) and $F2^*$ is the best known value of $F2$ from Eq. (25)). On the other hand, *part 2* finishes after *limit1* iterations without improving $F1^*$ or $F2^*$.

A table of frequencies is created during all of phase 2 (parts 1 and 2) to count the number of times that each order has been moved from one route to another. Every time that the end of *part 1* is reached, the average frequency for all the calls is calculated. During *part 2*, C is *fixed* and consists of all the calls whose frequency is below the average. Besides, in the insert moves of $x_i \in r_i$, $\delta = \max\{5, n\}$, where n is the number of orders in r_i. In *part 2*, no swap moves are executed.

The initial solution of *part 2* is the best known feasible solution defined by Eq. (24). If no feasible solution is known, the initial solution is the best known infeasible solution given by Eq. (25). The two important benefits of *part 2* are that it allows the effect of an inappropriate choice of δ to be corrected and it diversifies the search through the use of a *long term* memory.

3.2.2. Evaluation of the moves and aspiration criteria

This section answers simultaneously the questions of where to put the orders which are moved and how to evaluate the cost of a move in phase 2.

In the *insert* moves, if the order $x_i \in r_i$, with $r_i = [0, \ldots, x_h, x_i, x_j, \ldots, 0]$, is moved to the route r_k, the following change is made: the new route r_i is $r_i = [0, \ldots, x_h, x_j, \ldots, 0]$ and the new route r_k is the one generated by the GENI algorithm (with $p = 5$). An alternative, relative to r_i, would be to reorganise this route using the *US* algorithm (Gendreau et al., 1992), but this has not been implemented, because it appears that the computational time will not be compensated by a reduction in the cost of r_i.

In the *swap* moves, if the calls $x_i \in r_i$ and $x_j \in r_j$ are exchanged, the new routes r_i and r_j are con-

structed in two steps. First, x_i is removed from r_i and x_j from r_j as in the insert move. Second, x_i is inserted in r_j and x_j is inserted in r_i, both with the GENI algorithm.

In phase 2, the cost of a move is obtained from Eq. (25). Therefore, the *allowable* move with the lowest solution cost given by Eq. (25) is performed.

The solution value of a *feasible solution* is defined in this phase by Eq. (24), being $F1^*$ the minimum value. If no feasible solution is found $F1^*$ is considered to be infinite.

The two *aspiration criteria* for a move in phase 2 are either $F2 < F2^*$, or the solution is feasible and $F1 < F1^*$. When one or both of these conditions is satisfied, the tabu status of a move is overridden.

3.3. Phase 3 of the TSAMTVRSP

The initial solution of phase 3 is the best feasible solution from phase 2. So, phase 3 cannot be applied if no feasible solution to the problem can be found in phase 2. Phases 2 and 3 are very similar, the only two differences being that during the search a solution is never allowed to become infeasible relative to any constraint, and the solution value represents all the real variable costs, as given by Eq. (27):

$$F1 = \sum_{i=1}^{Nr} \beta \mathrm{Vc}_k d(i) + \sum_{i=1}^{Nr} \beta \mathrm{Cd}_k \{ t(i) + D\mathrm{Ot}(i) + \mathrm{OB}(D+1) \}, \quad (27)$$

where Vc_k is the cost/km of the vehicle type k, $d(i)$ is the length of the route i and Cd_k is cost/min for the driver of vehicle type k.

3.4. Tabu list size and tabu attributes

In the TSAMTVRSP, the set of trial moves is primarily dependent on the set of candidates, so that the tabu list size it is based on it. Since it has been shown experimentally by many researchers (for example, Taillard, 1991) that a tabu list of variable size tends to give better results than a fixed one, the tabu list size, θ, is taken to be a random number in the interval

$$\theta \in [\theta_{\min}, \theta_{\max}], \quad (28)$$

where

$$\theta_{\min} = \left\lceil \frac{n}{3c} \right\rceil, \quad (29)$$

or

$$\theta_{\min} = \left\lceil \frac{N}{3c} \right\rceil \quad (30)$$

and

$$\theta_{\max} = 2\theta_{\min}. \quad (31)$$

Expression (29) is applied in phase 1 and (30) in phase 2. In these equations, c is a parameter that the user can define. Since the best values of θ for each kind of problem and tabu search method have to be found empirically, computational tests are carried out with different real problems. Hence, the values tried were defined by taking $c = 2$.

In phase 1, the attribute used in the *insert* move is the order k which is moved. For the *swap* move, the attribute is the pair (i,j) of orders that are swapped. The restriction imposed in the *insert* move is that k cannot be moved again (by the insert move) during θ iterations. However, k can participate in a swap move and can change its position inside the route. The restriction in the *swap* move is to forbid another swap of (i,j) during θ iterations.

In the phase 2, the attribute used either in the *insert* move of k or in the *swap* move of the pair (i,j) is the route to which the calls (k or i or j) belong before the move. The restriction imposed in both types of move is that an order that leaves a route cannot return to the same route during the next θ iterations.

The presence of more than one route in a tour makes the algorithm (phases 2 and 3) much more complex, because when one route of the tour is modified, not only the cost of this new route has to be recalculated but the cost of the subsequent routes in the tour also has to be recalculated. Obviously, the routes have to be rescheduled before calculating the costs and this is the reason why the costs change.

4. Global description of the TSAMTVRSP algorithm

This section provides more operational details about the algorithm, putting together in sequence all the previous described operations which must be performed.

In phases 1 and 2, the routing is a matter of concern but the scheduling is more important, so the

solution value is measured in time units. This does not diminish the meaning and the quality of the solution for the company, because time is the factor with the strongest influence on the real costs. In phase 2, there is one function inside the GENI algorithm which is based on distance rather than on time. This is the function to determine the p nearest neighbours of each vertex. Therefore, the neighbourhood is defined in terms of distance but the evaluation of the solutions inside the GENI algorithm is based on time.

Because phase 3 is the last, it must be more precise in the solution evaluation than phases 1 and 2, so the solution value is a cost. The GENI procedure is more complex because all the costs are included in the objective function. The neighbourhood continues to be defined in terms of distance but the time is used to verify the feasibility of a solution, and the comparison between alternative feasible solutions is based on their variable costs.

A route is represented as $r = [0,1,\ldots,n,n+1]$, which shows the sequence in which the customers are visited, but does not determine the visiting hour. The *route schedule*, establishing both the sequence and the servicing time of each customer of the route, can be entirely determined employing the formulae of Section 2, after defining the departure time from the depot. For the first route of a tour, this is given by Eq. (32):

$$d_0 = \max\{e_0, e_1 - t_{01}\}. \tag{32}$$

For any other route r_i $(2 \le i \le h)$ of a tour $d_{0i} = a_{n+l_{i-1}} + u'_0$, where u'_0 is the time of loading the vehicle for the next route minus u_0 (the preparation of the vehicle in the morning), which is already included in the travelling time. The routes are all scheduled sequentially in the order they appear in the tour.

The departure time from a call i can be calculated as $d_i = \max\{a_i, e_i\}$, instead of using Eq. (9). Both ways are mathematically equivalent but this one has the advantage of avoiding calculating the waiting time, which is never needed for other purposes.

A route schedule is infeasible if any order (or the depot) has a negative forward slack (Sf).

The following describes the sequence of the main actions which constitute the TSAMTVRSP algorithm:

Step 1 (phase 1) – Do the following for each route separately:
- schedule the route;
- perform a set of moves according to the rules defined in Section 3.1. The execution of moves stops after a given number of iterations if the objective function (F from Eq. (22)) is not improved.

Step 2 (phase 2) – This phase consists of a given number of cycles. Each cycle is formed by *part 1* and *part 2*, starting with *part 1*.
Part 1
Execute a set of moves of the swap or insert type. After a given number (*limit*) of iterations (a move is executed in each iteration) without any improvement to either $F1$ or $F2$ (Eq. (24) and Eq. (25)) go to *part 2*.
Part 2
Execute moves of the insert type defined in Section 3.2. When the number of iterations without improving either $F1$ or $F2$ equals a given number (*limit1*), the search continues with *Step 3* if the number of stated cycles has been reached, or otherwise with *part 1*.

Step 3 (phase 3) – Execute a set of moves of the swap or insert type as defined in Section 3.2, allowing only the moves resulting in feasible solutions. The moves are evaluated by Eq. (27) and the execution stops when a given number of iterations has been executed without improving $F1$.

5. A real test of the TSAMTVRSP algorithm's performance

To test the TSAMTVRSP schedules were produced for vehicles operating from a distribution centre of the British company, Burton's Biscuits Ltd. Burton's fleet of vehicles consisted of 11 rigid vans and 11 tractor units with trailers, which have double the capacity of the vans. If these vehicles were insufficient to deliver all the orders, other vehicles could be hired. For Burton's vehicles the costs that appear in Eq. (27) were: $Vc_1 = £0.19/km$, $Vc_2 = £0.28/km$, $Cd_1 = £0.13/min$, $Cd_2 = £0.14/min$, where the subscript 1 is for vans and 2 for trailers. The tests were based on the orders delivered by

Burton's on ten consecutive working days. The schedules produced by an expert planner in the company were also obtained for each of these ten days. These schedules (referred to as manual schedules) contained the orders, the routing sequences and the vehicle identities. The manual schedules were not as detailed as the algorithm's schedules, in that the arrival, departure and waiting times at each call were in the algorithmic schedules but not in the manual ones. The manual schedules were converted to the algorithmic format using the rules of scheduling defined in Section 4. These rules minimise the number of orders of a given route with time window infeasibility.

The solutions given by the TSAMTVRSP depend on random factors (the set of candidates and the tabu list size), so different runs may produce different solutions. For each of the days, the programme was executed only once and the solutions obtained are given in Table 1. The parameters used were precisely those defined in Section 3. The algorithm automatically hired vehicles if the company had insufficient but none were needed during the ten days, either in the manual or in the algorithmic solutions.

Veh. infeasible is the number of orders that were delivered by a type of vehicle which in theory could not visit the particular calls, so that it does not appear in the algorithmic solution. *Dv. t. infeasible* is the number of tours that exceeded the maximum daily driving time of a driver (600 min), including overtime. *T. w. infeasible* is the number of routes that contained orders which were not delivered during the time windows. *Time* is the total travelling time of the schedules, including the waiting time. The *cost* of each schedule was calculated from Eq. (27). *T. best so.* is the CPU time taken to find the best solution on a Symmetry 2000. *CPU time* is the total CPU time taken from the beginning to end of the execution on the same computer.

Table 2 gives the percentage increases of the manual over the algorithmic solution, for each day.

It can be seen from Table 2 that the algorithmic solution is better than the manual solution on every measure. The fact that half of the manual solutions are infeasible widens further the difference in favour of the algorithmic solution, but the real influence of this cannot be quantified.

The multiple trips per day play an important role both in the manual solution and in the algorithmic one. There is no reason a priori for some vehicles to

Table 1
Manual and algorithmic solutions

Day		1	2	3	4	5	6	7	8	9	10
Number of orders		53	51	56	68	51	56	56	54	70	45
Manual solution	Veh. infeasible	0	0	0	0	0	0	0	2	0	0
	Dv.t.infeasible	0	0	0	2	0	0	0	0	0	0
	T.w. infeasible	0	2	0	1	1	0	0	2	2	0
	Time (min)	4847	5512	5631	5824	5027	5470	5545	4921	5691	4630
	Distance (km)	3136	3917	4196	4309	3768	3687	3840	3590	3893	3169
	Cost (£)	1423	1694	1713	1889	1560	1628	1623	1530	1735	1384
	Vans	8	7	9	9	9	9	10	8	8	6
	Trailers	8	11	11	8	9	7	7	10	11	8
	Routes	19	18	20	22	18	18	17	18	20	14
Algorithmic solution	Time (min)	4057	4796	4455	5050	4385	4215	3970	4437	4690	3552
	Distance (km)	2831	3417	3171	3591	3169	2940	2918	3219	3278	2679
	Cost (£)	1247	1498	1403	1559	1379	1414	1220	1397	1442	1229
	Vans	5	4	4	6	2	4	2	4	5	4
	Trailers	11	11	11	11	11	10	11	11	11	8
	Routes	16	16	15	18	14	14	13	15	17	12
	T. best so. (min)	13.4	15.8	22.1	12.6	19.0	12.4	7.8	9.3	16.4	11.6
	CPU time (min)	13.7	32.8	24.8	22.1	20.0	14.2	18.9	13.0	21.8	17.4

Table 2
Percentage increases of the manual over the algorithmic solutions

Day	1	2	3	4	5	6	7	8	9	10	Average
Time	19.5	14.9	26.4	15.3	14.6	29.8	39.7	10.9	21.3	30.3	22.3
Distance	10.8	14.6	32.3	20.0	18.9	25.4	31.6	11.5	18.8	18.3	20.2
Cost	14.1	13.1	22.1	21.2	13.1	15.1	33.0	9.5	20.3	12.6	17.4
Vehicles	0.0	20.0	33.3	0.0	38.5	14.3	30.8	20.0	18.8	16.7	19.2
Routes	18.8	12.5	33.3	22.2	28.6	28.6	30.8	20.0	17.6	14.3	22.7

undertake more than one route in the same day if vehicles of both types are left unused. What happens in the algorithmic solution is that some trailers make more than one trip, even though there are spare vans, because it is cheaper to deliver with the trailers or because one order exceeds the capacity of a van. The multiple trips are not used more frequently in the algorithmic solution because the number of orders is now low for the existing number of vehicles. The situation was completely different only three years previously, when the research started. At that time, Burton's Biscuits delivered around 200 orders per day from the distribution centre to the individual retail outlets of the major supermarket chains. Since then, large orders have increasingly been delivered into the central warehouses of the leading retailers and that process will be complete in 1997, when Asda is expected to cease having biscuits delivered to individual stores. Whilst the situation was still evolving, Burton's Biscuits did not wish to change from manual load planning.

Strict comparisons between different real problems for the gains obtained from using algorithmic instead of manual solutions cannot be made, because of the intrinsic differences between problems and the consequential difficulties experienced by even the most skilful manual load planners. Nevertheless the improvements achieved by applying the TSAMTVRSP algorithm, which averaged about 20% better than the manual solutions of a very experienced planner, merit examination in the context of others reported in the literature. Vliet et al. (1992) report an improvement of 7% over the manual solution. The gains reported by Semet and Taillard (1993) and by Potvin et al. (1990) are 10–15% and 10% respectively, over an existing heuristic. Another case which is probably more representative than those three, in that the problem characteristics are more similar to the one studied here, is due to Koskosidis and Powell (1990). Their improvements over the manual solution vary from 0.07% to 8.55% but there is violation of the time windows in both the algorithmic and the manual solutions.

6. For further reading

Glover, 1977

Acknowledgements

The authors thank Burton's Biscuits' staff for providing the company data and giving freely of their time. The Ordnance Survey's permission to use the digital map data is gratefully acknowledged. The first author's research was partially supported by a grant from Junta Nacional de Investigação Científica e Tecnológica. We also thank two unknown referees for their most constructive comments.

References

Barnes, J.W., and Laguna M. (1993), "A tabu search experience in production scheduling", in: F. Glover, M. Laguna, E. Taillard and J.C.D. de Werra (eds.), *Annals of Operations Research* 41, 141–156, Baltzer AG, Science Publishers, Basel, Switzerland.

Brandão, J.C.S. (1994), *A Decision Support System and Algorithms for the Vehicle Routing and Scheduling Problem*, Doctor of Philosophy Thesis, Department of Management Science, Lancaster University.

Gendreau, M., Hertz, A., and Laporte, G. (1992), "New Insertion and post-optimization procedures for the traveling salesman problem", *Operations Research* 40/6, 1086–1094.

Gendreau, M., Hertz, A., and Laporte, G. (1994), "A tabu search heuristic for the vehicle routing", *Management Science* 40, 1276–290.

Gendreau, M., Hertz, A., Laporte, G., and Stan, M. (1996), "Ageneralized insertion heuristic for the traveling salesman problem with time windows", Publication CRT-95-07, Centre de Recherche Sur les Transports, Université Montreal.

Glover, F. (1977), "Heuristics for integer programming using surrogate constraints", *Decision Sciences* 8, 156–166.

Glover, F. (1989), "Tabu search – Part I", *ORSA Journal on Computing* 1/3, 190–206.

Glover, F. (1990), "Tabu search – Part II", *ORSA Journal on Computing* 2/1, 4–31.

Glover, F., Taillard, E, and de Werra, J.C.D. (1993), "A user's guide to tabu search", in: F. Glover, Laguna, E. Taillard and J.C.D. de Werra (eds.), *Annals of Operations Research* 41, 3–28, Baltzer AG, Science Publishers, Basel, Switzerland.

Kelly, M.J., Golden, B., and Assad, A. (1993), "Large-scale controlled rounding using tabu search with strategic oscillation", in: F. Glover, M. Laguna, E. Taillard and J.C.D. de Werra (eds.), *Annals of Operations Research* 41, 69–84, Baltzer AG, Science Publishers, Basel, Switzerland.

Koskosidis, Y.A., and Powell, W.B. (1990), "Application of optimization based models on vehicle routing and scheduling problems with time windows constraints", *Journal of Business Logistics* 11/2, 101–128.

Lenstra, J.K., and Rinnooy Kan, A.H.G. (1981), "Complexity of vehicle routing and scheduling problems", *Networks* 11, 221–227.

Mooney, E.L., and Rardin R.L. (1993), "Tabu search for a class of scheduling problems", in: F. Glover, M. Laguna, E. Taillard and J.C.D. de Werra (eds.), *Annals of Operations Research* 41, 69–84, Baltzer AG, Science Publishers, Basel, Switzerland.

Potvin, J., Lapalme, G., and Rousseau, J. (1990), "Integration of AI and OR techniques for computer-aided algorithmic design in the vehicle routing domain", *Journal of the Operational Research Society* 41, 517–525.

Savelsbergh, M.W.P. (1988), *Computer Aided Routing*, Doctoral Dissertation, Centrum voor Wiskunde en Informatics, Amsterdam.

Rochat, Y., and Semet, F. (1994), "A tabu search approach for delivering pet food and flour in switzerland", *Journal of the Operational Research Society* 45, 1233–1246.

Semet, F., and Taillard E. (1993), "Solving real-life vehicle routing problems efficiently using tabu search", in: F. Glover, M. Laguna, E. Taillard and J.C.D. de Werra (eds.), *Annals of Operations Research* 41, 469–488, Baltzer AG, Science Publishers, Basel, Switzerland.

Taillard, E. (1991), "Robust tabu search for the quadratic assignment problem", *Parallel Computing* 17, 443–455.

Taillard, E., Laporte, G., Gendreau, M. (1996), "Vehicle routing with multiple use of vehicles", *Journal of the Operational Research Society* 47, 1065–1070.

Vliet, A., Boender, C.G.E., and Rinnooy Kan, A.H.G. (1992), "Interactive optimization of bulk sugar deliveries", *Interfaces* 22/3, 4–14.