Journal of the Operational Research Society (1998) 49, 799–805
http://www.stockton-press.co.uk/jor

The multi-trip vehicle routing problem

JCS Brandão[1] and A Mercer[2]
[1]*Universidade do Minho, Braga, Portugal and* [2] *Lancaster University*

The basic vehicle routing problem is concerned with the design of a set of routes to serve a given number of customers, minimising the total distance travelled. In that problem, each vehicle is assumed to be used only once during a planning period, which is typically a day, and therefore is unrepresentative of many practical situations, where a vehicle makes several journeys during a day. The present authors have previously published an algorithm which outperformed an experienced load planner working on the complex, real-life problems of Burton's Biscuits, where vehicles make more than one trip each day. This present paper uses a simplified version of that general algorithm, in order to compare it with a recently published heuristic specially designed for the theoretical multi-trip vehicle routing problem.

Keywords: vehicle routing; multiple trips; tabu search

Introduction

The basic *vehicle routing problem* (VRP) can be defined as follows. Given a depot and a number of customers with known geographical locations and demands, the VRP consists of finding a set of routes that minimises the total distance travelled, whilst satisfying all the customers' requirements. A route, or trip, consists of a sequence of visits to customers, starting and ending at the depot. The total demand of all the customers on a route must not exceed the vehicle capacity. Each route, which has limited length or duration, is assigned to one vehicle, all of which are identical, and each customer is visited exactly once.

In an empirical study for Burton's Biscuits Ltd., Hommes *et al*[1] compared a number of load planning algorithms for the real operations at a regional distribution centre. Using a development of the method employed by the company, all capacity constrained vehicles expected to return to the depot within seven hours were considered for a second trip. If necessary, the time available for the second trip was increased by removing orders from the first trip, provided the total quantity delivered on the two trips exceeded the quantity originally scheduled for the first. Their simulation showed that the number of vehicles could be reduced from 21–19 and the unit cost of the deliveries was 5% less. Subsequently, Brandão[2] addressing again the distribution system of Burton's Biscuits Ltd., concluded that multiple trips (MT) were most appropriate for the company's VRP, leading to his MTVRP formulation. That MTVRP algorithm was used on a practical problem of Burton's Biscuits and took into account the company's real distribution requirements: any number of multiple trips per day; delivery time windows; different types of vehicle, the hiring of vehicles if the company has insufficient on any day; restricted access to some customers for certain types of vehicles; legal limits on the drivers' schedules. Moreover, distances were determined from the road network and the objective function included the costs of drivers, fuel, fleet maintenance and hired vehicles. Brandão and Mercer[3] gave the details of the algorithm and the actual company data used in the calculations. These showed that the algorithm outperformed the manual schedules of a very experienced load planner by about 20% but no alternative heuristic, with which comparisons could be made, was available when Brandão[2] performed the calculations for Burton's Biscuits in 1994.

Fleischmann[4] has also studied the MTVRP problem and suggested using bin packing in the final part of the algorithm. More recently, Taillard *et al*[5] have published an algorithm, which starts by generating a large set of routes satisfying the VRP constraints and finishes by assembling selected routes with a bin packing heuristic into feasible working days. None of the practical features essential for the real applications, such as having delivery time windows and vehicles of different types were included by Taillard *et al*[5] but their heuristic does at least provide a very different approach with which to assess the algorithm of Brandão and Mercer.[3] However, before any comparisons can be made, the algorithm developed for the Burton's Biscuits problem has had to be simplified by the removal of all its practical features, except for the multiple trips and the maximum driving time per day. The objective function of this simplified algorithm (named TSMTVRP) has necessarily been changed to one of minimising the total travelling time for a given number of vehicles.

Correspondence: Dr JCS Brandão, Departamento de Gestão e Administração Pública, EEG, Universidade do Minho, 4709 Braga codex, Portugal.
E-mail: sbrandao@ci.uminho.pt

The TSMTVRP is described in the following section and then compared with the algorithm of Taillard *et al*[5] using severn problems taken from Christofides *et al*[6] and two problems taken from Fisher.[7]

The algorithm

The final solution for the MTVRP is given by the TSMTVRP, which is a tabu search algorithm composed of two phases. It starts from an initial solution given by an algorithm called NENBMTVRP, which is an algorithm combining nearest neighbour and insertion concepts.

Nomenclature

Tour = a set of routes assigned to the same vehicle.
N = total number of customers.
V = total number of vehicles, which may be defined a priori or determined by using the algorithm.
Q = maximum capacity of each vehicle.
T = maximum driving time.
To = maximum allowable driver's overtime.
D = a constant which penalises the use of overtime. When $D = 1$, as assumed here, overtime costs twice the normal time.
Nr = total number of routes in the solution.
$n(r)$ = number of customers on route r.
$t(r)$ = travelling time of route r.
h_v = number of routes assigned to vehicle $v, v = 1, 2, \ldots, V$.
Tr_v = remaining normal driving time for vehicle v, when each driver is assigned to one vehicle, so that

$$Tr_v = T - \sum_{r=1}^{h_v} t(r)_v. \tag{1}$$

Ot_v = overtime for vehicle v, given by

$$Ot_v = \max\left(0, \sum_{r=1}^{h_v} t(r)_v - T\right) \tag{2}$$

Te_v = length of time by which Ot_v exceeds To, given by

$$Te_v = \max(0, Ot_v - To). \tag{3}$$

δ-neighbourhood of a customer: set of the δ nearest neighbours of this customer, with the value of δ based on the expected number of customers per route.

Admissible: a customer is admissible to a route if Q and To are not exceeded.

NENBMTVRP

There are two independent procedures within NENBMTVRP: nearest neighbour and insertion. Firstly, the nearest neighbour procedure is applied to create a *layer*, which is a set of routes such that each route belongs to a different tour (vehicle). Then the insertion procedure is applied to the routes of this layer to try to include more customers. Using the nearest neighbour and insertion procedures to create a layer is called a *stage*. The final solution given by the NENBMTVRP results from successively adding layers by stages.

The maximum number of routes per layer is V but it will be less than V if Tr of one or more vehicles is insufficient for the addition of more routes or if all the customers have already been routed. At the beginning of each stage, the vehicles are put on a list in descending order of Tr. Then a route is started with the unrouted customer farthest from the depot. It is assigned to the first vehicle of this list that has not yet received a route in this stage and where Tr is long enough for the route. If no such vehicle exists, the number of routes in the layer is complete and the insertion procedure begins. After a route has been started, its construction proceeds as follows. Among the unrouted customers that belong to the δ-neighbourhood of each customer in the route, choose the admissible one which increases the routing time least. This routing time is calculated assuming that the new customer must be inserted immediately before or after the customer in whose δ-neighbourhood it belongs. Obviously, a customer can be inserted near another only if it belongs to its δ-neghbourhood. If no customer is admissible to enter the route, a new route is started.

The insertion procedure consists of inserting unrouted customers in the routes of the last layer produced by using the nearest neighbour procedure. The choice of the unrouted customer to enter a route is made in the following way. For each unrouted customer, its insertion cost (time) is calculated for each of the routes of the layer where this customer is admissible, trying the insertion near each customer of the route. The customer with the least insertion cost is inserted and the process is repeated until no more customers are admissible. It is possible to use overtime but only for the routes of the last layer of each vehicle if there are still unrouted customers that could not be inserted using the prescribed maximum driving time.

TSMTVRP algorithm

This tabu search algorithm consists of two phases which are applied sequentially. The difference between them is that a solution is allowed to become infeasible in phase 1 in relation to the driving time but only feasible solutions are accepted in phase 2. The initial solution of phase 2 is the best feasible solution from phase 1.

This algorithm contains several parameters, whose values were originally determined intuitively from experience with other, similar algorithms. The values have since been refined by computational experimentation, including during the research on the Burton's Biscuits problem. Those used in the present calculations perform well,

although it may be that a better combination could be found.

The solution at each iteration of the tabu search consists of a set of routes Nr. A set of customers C_r with cardinal given by (4) is chosen randomly from each route r

$$C_r = \left\lceil \frac{n(r)}{c} \right\rceil. \tag{4}$$

Taking c to be 2 has been found to give good results.

Joining the set of customers of each route produces the *set of candidates* C, given by (5)

$$C = \bigcup_{r=1}^{Nr} C_r. \tag{5}$$

There are two kinds of trial moves in TSMTVRP: *insert* and *swap* moves.

The *insert* move consists of removing one customer from one route (origin) and putting it into another route (destination), the destination route can be an existing route or a new route.

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

(i) at least one customer of r_j belongs to the δ-neighbourhood (δ nearest neighbours) of x_i, or r_j is a new route,
(ii) the insertion of x_i in r_j does not cause an excess of load in the vehicle that contains r_j.

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

(i) at least one customer of r_j belongs to the δ-neighbourhood of x_i and at least one customer of r_i belongs to the δ-neighbourhood of x_j,
(ii) the exchange of x_i and x_j does not cause an excess of load either in r_i or r_j,
(iii) at least one of the routes r_i or r_j contains more than one customer.

The parameter δ is a way of limiting the number of potential moves and it is unlikely to have an appreciable negative effect on the quality of the solutions, if any. Initially $\delta = 5$, but it is increased after a given number of iterations.

The rule (i) 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.

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 and the overtime, the same factors as in F1 are included in F2 plus one to evaluate the infeasibility

$$\mathrm{F1} = \sum_{i=1}^{Nr} t(i) + \sum_{v=1}^{V} D \times Ot_v, \tag{6}$$

$$\mathrm{F2} = \sum_{i=1}^{Nr} t(i) + \sum_{v=1}^{V} (D \times Ot_v + P \times Te_v). \tag{7}$$

The penalty P is initially taken to be 1. P is multiplied by 2 if all the solutions have been infeasible during α consecutive iterations. Conversely, P is divided by 2 if all the solutions have been feasible during the last α consecutive iterations. A good starting value is $\alpha = 10$. Using P to create strategic oscillation has been applied successfully by such authors as Kelly *et al*,[8] Mooney and Rardin[9] and Gendreau *et al*,[10] who have varied P in a VRP algorithm just like in the present algorithm.

Part 1 is replaced by part 2 when a given number of iterations ($limit = 4N$) has been executed without improving either F1* or F2*, where F1* is the best known value of F1 from (6) and F2* is the best known value of F2 from (7). On the other hand, part 2 finishes after $limit1 = \lceil 1.5N \rceil$ iterations without improving F1* or F2*.

A table of frequencies is created to count the number of times that each customer 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. Additionally, in the insert moves of $x_i \in r_i$, $\delta = \max\{5, n\}$, where n is the number of customers in r_i. In part 2, no swap moves are executed.

The initial solution of part 2 is the best feasible solution from (6). If no feasible solution is available, the initial solution is the best known infeasible solution given by (7). The two important benefits derived from part 2 are that the effect of an inappropriate choice of δ is corrected and the search is diversified through the use of a *long term* memory.

Evaluation of the moves and aspiration criteria

In the *insert* moves, if the customer $x_i \in r_i$, 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 familiar GENI algorithm (Gendreau *et al*[11]), with $p = 5$. When the number of customers in r_i' exceeds nine, this route is reorganised using the *US* algorithm (Gendreau *et al*[11]), in order to reduce the cost of r_i'. In general, both the solution quality and the computing time increase with the parameter p of the GENI algorithm, so that its originators recommend a value of 5 as being a good compromise.

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 constructed in two

steps. Firstly, x_i is removed from r_i and x_j from r_j as in the insert move. Secondly, x_i is inserted in r_j and x_j is inserted in r_i, both with the GENI algorithm.

In phase 1, the cost of a move is obtained from (7). Therefore, the *allowable* move with the lowest solution cost given by that equation is made. The solution value of a *feasible solution* is defined by (6), being the minimum value F1*. If no feasible solution is obtained F1* is considered to be infinite.

The two *aspiration criteria* for a move in phase 1 are either F2 < F2*, or the solution is feasible and F1 < F1*. When at least one of these conditions is satisfied, the tabu status of a move is overridden. The only *aspiration criterion* in phase two is that F1 < F1*.

The tabu list size and tabu attributes

The set of trial moves in the TSMTVRP is dependent primarily on the set of candidates, so the tabu list size is based on it. Since it has been shown experimentally by researchers such as Taillard[12] 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 $\theta \in [\theta_{\min}, \theta_{\max}]$, where $\theta_{\min} = \lceil N/3c_1 \rceil$, and $\theta_{\max} = 2\theta_{\min}$. The parameter c_1 may be defined by the user, since the best value of θ for each type of problem and tabu search method have to be found empirically. Values defined by taking $c_1 = 2$ were used in this application.

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 during both types of move is that a customer leaving a route cannot return to the same route during the next θ iterations.

The sequence of the TSMTVRP operations

More operational details are provided here about the algorithm, sequencing all the necessary operations described previously.

Each phase consists of a given number of cycles defined by the user. In the problems solved, two cycles have been used. Each cycle is formed by part 1 and part 2, starting with part 1. There are two iteration counters; one that counts the total number of iterations performed inside each phase (the execution changes phase or stops when the maximum allowed is reached) and another that counts the total number of iterations performed without improving either F1 or F2. Hence, this second counter is set equal to zero every time there is an improvement in either F1 or F2 or in both, when there is a change from part 1 to part 2 and inside part 1 at its restart. The maximum number of iterations associated with the first counter was limited to $80N$, but the limit is not normally reached, because the two cycles are completed more quickly. Therefore there are two stopping rules based on the number of iterations executed: a deterministic upper limit and a stochastic rule depending on the search process described below.

Part 1: Exeute a set of moves of the swap or insert type, as defined previously. After a given number ($limit = 4N$) of iterations (a move is executed in each iteration) without any improvement to either F1 or F2, go to part 2 or repeat part 1. Part 1 consists of three subcycles and only after performing them does the program start part 2. In the first subcycle $p = 5$ and $\delta = 5$, and in each following subcycle p is increased by one unit and δ by five units. On the transition to part 2, p and δ are restored to 5.

Part 2: Execute only moves of the insert type. When the number of iterations without improving either F1 or F2 equals a given number ($limit1 = \lceil 1.5N \rceil$), the search continues with phase 2, if the stated number of cycles has been reached, or otherwise with part 1.

Computational complexity of the TSMTVRP

The analysis of computational complexity is necessarily based on the complexity of each iteration, but it is impossible to know a priori the number of trial moves in each iteration. This will depend on the number of routes, the number of customers per route, the δ-neighbourhood and even the demand of each customer (see condition (ii) of the trial moves). However, a reasonable approximation is to assume that every route has the same number of customers (n) so that the number of routes is approximately N/n. Therefore the number of trial insert moves per iteration is $n/2 \times N/n = N/2$ and the number of trial swap moves is given by $[(N-n)/n \times N/2n]\,(n/2)^2 = (N-n)N/8$.

The insertion of one customer in a route with the GENI algorithm has computational complexity $O(p + p^3 + p^4)$. Therefore the overall complexity of each iteration is approximately $O([N-n)N/8 + N/2]\,(p + p^3 + p^4])$. Therefore the value of p has a major influence on the computing time and the swap moves account for most of this time when N is much larger than n. Therefore, the average computing time per iteration can be reduced substantially if the swap moves are not performed every iteration but, for example, every five iterations. The consequence of this, as verified by some computational experiments, is that the solution quality tends to be a little worse, on average.

The programme, as written, makes use of a memory structure that allows a substantial reduction to be made to the above complexity. It consists of retaining information from one iteration to another, during which only two routes will have changed, so that the existing information about all the other potential transformations remains unaffected.

Computational comparisons

Taillar *et al*[5] illustrated their algorithm with calculations for problems, which include 1–5 and 11 and 12 taken from

Christofides *et al*,[6] and 11 and 12 taken from Fisher,[7] so that these have been used to make comparisons with the TSMTVRP algorithm. The same number of vehicles, V, and the same maximum driving times, T_1 and T_2, have been used, where the values of T_1 and T_2 are given by $1.05Z^*/V$ and $1.1Z^*/V$, respectively, rounded to the nearest integer. Therefore each of the values of V, being the integers between one and the maximum number of vehicles (V_{max}), provides a subproblem. The values of Z^* are the quasi-optimal solution of the VRP with no limit on the number of vehicles used, taken from Rochat and Taillard.[13] All these problems are Euclidean and it is assumed that the travelling time between each pair of customers is numerically equal to the corresponding distance.

The TSMTVRP has been programmed in C and run on a HP Vectra XU Pentium Pro at 200 MHz, which according to Dongarra[14] is 2.5 times faster than the 100 MHz Silicon Graphics Indigo workstation used by Taillard *et al*.[5] This number is based on tests benchmarked with a program to solve standard linear equations, LINPACK, measuring the number (in millions) of floating-point operations per second (Mflop/s) executed by each computer. Dongarra[14] reported a speed of 38 Mflop/s for the Pentium Pro 200 MHz and 15 Mflop/s for the SGI Indigo2 Extreme (R400/100 MHz). Our first difficulty in making a fair comparison is that the models used for the algorithms may not be exactly the same as in the tests. However, the main difficulty comes from the fact that the software and even the measure, may be unrepresentative for the relative speed of the two algorithms. Dongarra[14] recognises this when he states that 'the timing information presented here should in no way be used to judge the overall performance of the computer system'. Nevertheless, in the absence of better information, that ratio is assumed to be valid.

Table 1 gives the problem number and the total number of customers of Christofides *et al*[6] and Fisher[7] (prefixed, respectively, with letters C and F) and the value of V_{max}, for which Taillard *et al*[5] give computational results. In this table, I identifies the subproblem for which the solution obtained is infeasible. The CPU time for Tailard *et al*[5] is their stated average over all runs. The value given for the TSMTVRP algorithm is the average computing time taken to find a feasible solution for each problem number, considering only those instances for which a feasible solution was found. The *relative* CPU time is the TSMTVRP value multipled by the 2.5 to allow for the faster speed of the Pentium Pro 200 MHz computer. These times must be treated with some caution, not least because Taillard *et al*[5] noted that they obtained similar results if about half their stated computing times were used but it does appear from the published information that the TSMTVRP algorithm is faster.

Like many algorithms TSMTVRP depends on some random parameters, so that the solution obtained may vary from one execution to another but the differences are invariably small. TSMTVRP was executed at most five times for each problem, but it was executed only once in most cases, because a feasible solution was found at the first attempt. The infeasible solutions presented in Table 2 are the best of the five executions in terms of routing cost.

The TSMTVRP always produced feasible solutions for T_2, compared with the four infeasible results of Taillard *et al*.[5] For T_1, the number of infeasible solutions to the 52 subproblems are 15 and 14 for the TSMTVRP and the

Table 1 Computational results

				Taillard et al[5]			*TSMTVRP*		
Problem number	N	V_{max}	V	T_1	T_2	*CPU time (min)*	T_1	*CPU time (min)*	*Relative CPU time (min)*
C1	50	4	3	I	I	5	I	1	2.5
			4	I			I		
C2	75	7	6	I		7	I	2	5
			7	I	I		I		
C3	100	6	5	I		24		4	10
			6	I			I		
C4	150	8	7	I		51	I	10	25
			8	I	I		I		
C5	199	10	9			66	I	25	62.5
			10	I			I		
C11	120	5	4	I		45	I	10	25
C12	100	6	4			23	I	4	10
			5	I			I		
			6	I			I		
F11	71	3	2	I		26	I	1	2.5
			3	I	I		I		
F12	134	3				75		32	80

Table 2 Comparison of routing times

Problem number	V	Z^*	*Taillared et al*[5]			*TSMTVRP*			*Difference (%)*		
			Time	*Cost*	*Ratio*	*Time*	*Cost*	*Ratio*	*Time*	*Cost*	*Ratio*
C1	3	524.61	533.00	579.48	1.115	556.34	575.73	1.041	4.38	−0.65	−6.64
	4		546.29	565.27	1.027	547.10	564.07	1.027	0.15	−0.21	0.00
C2	6	835.26	841.60	857.19	1.032	858.04	867.02	1.031	1.95	1.15	−0.10
	7		843.60	878.29	1.073	870.07	896.57	1.088	3.14	2.08	1.40
C3	5	826.14	829.50	853.23	1.062	845.89	845.89	0.999			
	6		842.85	861.18	1.032	837.82	838.74	1.003	−0.60	−2.61	−2.81
C4	7	1028.42	1042.39	1074.16	1.033	1063.95	1102.63	1.071	2.07	2.65	3.68
	8		1049.02	1088.03	1.075	1069.54	1085.94	1.031	1.96	−0.19	−4.09
C5	9	1291.44				1329.28	1359.75	1.056			
	10		1316.00	1331.09	1.024	1336.92	1359.75	1.051	1.59	2.15	2.64
C11	4	1042.11	1042.11	1055.07	1.020	1069.24	1075.35	1.011	2.60	1.92	−0.88
C12	4	819.56				819.56	825.94	1.012			
	5		819.56	836.80	1.050	828.94	841.14	1.036	1.14	0.52	−1.33
	6		819.56	845.48	1.064	819.56	847.85	1.072	0.00	0.28	0.75
F11	2	241.97	241.97	249.97	1.031	255.12	257.98	1.011	5.43	3.20	−1.94
	3		244.60	257.31	1.075	254.40	257.47	1.011	4.01	0.06	−5.95
								Average	2.14	0.80	−1.18

algorithm of Taillard *et al*,[5] respectively. The infeasible solutions with a maximum driving time of T_1 have a total travelling time for all the vehicles of less than $1.05Z^*$, except for three of the TSMTVRP results, indicating that the infeasibility is due to an imbalance between the times of vehicles on multi-trip routes.

Table 2 contains, for both algorithms and those subproblems when either produced an infeasible solution with respect to T_1, the total routing *time*, the *cost* assuming that overtime costs three times the normal ($D = 2$ in (6)) and the *ratio* between the longest tour of each solution and T_1. As the values for the two feasible solutions of Taillard *et al*[5] are unknown, the percentage differences between the corresponding values of each solution and the averages given in Table 2 are for the 13 subproblems for which both algorithms failed to obtain feasible solutions.

These results show that by comparison with the results of Taillard *et al*,[5] TSMTVRP averages 2.14% longer for the routing times and 0.80% higher for the cost but the ratio is −1.18% lower. Therefore although Taillard *et al*[5] algorithm gives solutions slightly better in terms of routing time, it is less efficient at balancing daily tours. This follows because the average cost difference is smaller than the average time difference and the ratio is even lower, in spite of the global routing time being longer. The following four examples, for which the algorithm Taillard *et al*[5] was unable to find feasible solutions to corroborate this conclusion. The routing times of Taillard *et al*[5] for the problems, with T_2, C1 ($V=3$), C2 ($V=7$), C4 ($V=8$) and F11 ($V=3$) are 529.17, 843.60, 1044.98 and 244.60 respectively, compared to the corresponding TSMTVRP values of 573.91, 878.85, 1104.85 and 258.48. The fact that TSMTVRP routing times are longer is of secondary importance compared with the feasibility of the solution, as the execution of the algorithm was stopped as soon as a feasible solution was obtained.

Conclusions

An algorithm, TSMTVRP, has been produced by simplifying one described previously in Brandão and Mercer.[3] The changes to the latter, which was developed to minimise the total costs of a real problem of Burton's Biscuits, were made in order to compare the algorithm with a totally different approach of Taillard *et al*.[5] Their heuristic, which does not include delivery time windows or vehicles of different types, obtains the best solutions for the VRP and then combines these solutions using a bin packing algorithm, in order to obtain a feasible solution for the MTVRP. Even though their heuristic was designed specifically for the basic theoretical problem, the routing times of Taillard *et al*[5] are only very slightly less than those from the algorithm of Brandão and Mercer,[3] which is a simplified version of one that has been used on a real problem, with all its practical complications. Moreover, the Taillard *et al*[5] heuristic appears to be slower and does not balance the tours as well as TSMTVRP. For those reasons but also taking into account the structures of the two approaches, we question the claim of Taillard *et al*[5] that their approach could easily be applied to solve real-life problems and, in any event, we expect it to be inferior to the approach of Brandão and Mercer.[3]

Acknowledgements—The authors are indebted to the referees for their valuable comments on this article.

References

1 Hommes P, Howe ER and Pape CS (1989). Burton's Biscuits: a study of the load planning operation at the new depot at Risley. M.Sc. Dissertation, Lancaster University.

2 Brandão J (1994). A decision support system and algorithms for the vehicle routing and scheduling problem. Ph.D. Thesis, Lancaster University.

3 Brandão J and Mercer A (1997). A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *Eur J Opl Res* **100**: 180–191.

4 Fleischmann B (1990). The vehicle routing problem with multiple use of vehicles.Working paper, Fachbereich Wirtschaftswissenschaften, Universität Hamburg.

5 Taillard E, Laporte G and Gendreau M (1996). Vehicle routeing with multiple use of vehicles. *J Op Res Soc* **47**: 1065–1070.

6 Christofides N, Mingozzi A and Toth P (1979). The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P and Sandi C (eds). *Combinatorial Optimization*. Wiley: Chichester, pp 313–318.

7 Fisher ML (1994). Optimal solution of vehicle routing problems using minimum k-trees. *Opns Res* **42**: 626–642.

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

9 Mooney EL and Rardin RL (1993). Tabu search for a class of scheduling problems. In: Glover F, Laguna M, Taillar E and Werra JCD (eds). *Annals of Operations Research* **(41)**. Baltzar AG, Science Publishers: Basel, Switzerland, pp 3–28.

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

11 Gendreau M, Hertz A and Laporte G (1992). New insertion and post-optimization procedures for the traveling salesman problem. *Opns Res* **40**: 1086–1094.

12 Taillard E (1991). Robust tabu search for the quadratic assignment problem. *Parallel Comput* **17**: 443–455.

13 Rochat Y and Taillard E (1995). Probabilistic diversification and intensification in local search for vehicle routing. *J Heuristics* **1**: 147–167.

14 Dongara JJ (1998). Performance of various computers using standard linear equations software. Report CS-89-85, University of Tennessee.

Received May 1997;
accepted April 1998 after two revisions