Local Search Strategies for the Vehicle Fleet Mix Problem

Ibrahim H. Osman and Said Salhi

Abstract

The vehicle fleet mix (VFM) problem is the vehicle routing problem with additional practical extensions. The VFM problem involves a heterogeneous fleet of vehicles with their associated variable running and fixed purchasing/renting costs. The objective is to find the optimal fleet composition of vehicles and a set of feasible routes that minimize the total costs. In this paper, two techniques are proposed: a constructive heuristic and a tabu search metaheuristic. The constructive heuristic is an enhanced modification of the Salhi and Rand route perturbation procedure. The tabu metaheuristic is new and it uses a compound-moves neighbourhood with a special data structure. Computational results are reported on a set of 20 test problems from the literature. The proposed methods obtain new results that improve upon the best published results.

8.1 Introduction

The vehicle fleet mix (VFM) problem is the vehicle routing problem with a heterogeneous fleet of vehicles. Each vehicle is characterized by the carrying capacity, maximum travel time, variable running cost and fixed purchasing/renting cost. Each vehicle route originates and terminates at a central depot in order to service a set of customers with known demands. The total cost involved in the VFM problem consists of the fixed vehicle utilization costs as well as the variable routing and scheduling costs. The later costs include the distance and travel time costs in addition to the service (loading/unloading) time costs. The constraints of the VFM problem are: (i) each customer must be supplied by exactly one vehicle route; (ii) the total load carried by any vehicle must not exceed its maximum capacity; (iii) the total length of any route, which includes the inter-customer travel times and service times, must not exceed a pre-specified limit. The objective is to find the optimal mix of heterogeneous vehicles and the associated set of feasible routes that minimizes the total variable and fixed

Modern Heuristic Search Methods
Editor V. J. Rayward-Smith, I. H. Osman, C. R. Reeves and G. D. Smith©1996 John Wiley & Sons Ltd.

costs.

The VFM problem is practically more important than its counterpart, the vehicle routing problem (VRP). The VRP, however, has attracted much more research attention. The lack of VFM research is mainly due to the less obvious way of how to determine the best mix of vehicles which is a medium-term decision problem. The decision problem involves an important cost investment factor. If an unwise decision was made to purchase a fleet of vehicles, it would be difficult to change such a decision. In general, researchers have considered a homogeneous fleet of vehicles. Yet in real life, the appropriate fleet is, by no means, homogeneous and a heterogeneous vehicle fleet is likely to yield better results. Furthermore, the VFM problem still has the difficulty of the operational routing decision which is encountered in the daily operations of vehicles in the classical VRP. Hence, the VFM problem is computationally more complex than the VRP which is known to be NP-hard (Lenstra and Rinnooy Kan[LR81]). Consequently, an exhaustive search of the solution space is impossible for VFM problems of large size and there is no known algorithm that can solve the VFM problem to optimality in a polynomial time. The complexity of the VFM problem therefore necessitates the development of effective heuristics that are capable of providing high-quality approximate solutions within a reasonable amount of computational effort.

Local search methods form a class of heuristics that proceed by examining some neighbourhoods of the current solution. The simplest type of local search method is the descent algorithm. It starts with any solution, neighbours are searched until an improved solution is found. A further search is then initiated from the improved solution and the algorithm continues until no further improvement is possible. The solution obtained may deviate significantly from the optimum. Tabu search (TS) is a metaheuristic which is superimposed on the descent algorithm to guide the search process to avoid getting trapped in bad local optima. TS allows the search to proceed to a neighbour even if it causes a deterioration in the objective function value. TS has been proven to be a very effective and successful metaheuristic for solving many combinatorial optimization problems. Particularly impressive results have been obtained for many vehicle routing problems: Eglese and Li[EL96], Gendreau, Hertz and Laporte[GHL94], Potvin, Kervahut, Garcia and Rousseau[PKGR96], Osman[Osm93], Rego and Roucairol[RR96], Salhi and Sari[SS95], Taillard[Tai93], Thangiah, Osman and Sun[TOS94] and Thangiah, Osman, Vinayagamoorthy and Sun[TOVS93]. For recent bibliographies, we refer to Osman and Laporte[OL96] on metaheuristics, to Laporte and Osman[LO95] on routing problems. For an overview and an introduction on metaheuristics, we refer to Osman[Osm95b] and Osman and Kelly[OK96b]. For metaheuristic books, we refer to: Aarts and Lenstra[AL96], Laporte and Osman[LO96], Osman and Kelly[OK96a], Rayward-Smith[RS95] and Reeves[Ree93]. Motivated by tabu search successes, in this paper, we present a tabu search algorithm for solving the mixed fleet vehicle routing problem. We design a special data structure to evaluate efficiently trial solutions generated by the 1-interchange (compound-moves) neighbourhood mechanism[Osm95a, Osm93, OC94]. Finally, we introduce some modifications and refinements to enhance the performance of the RPERT procedure proposed by Salhi and Rand [SR93] for the VFM problem.

The paper is organized as follows. The vehicle fleet mix and its relevant literature are reviewed in Section 8.2. The modified RPERT construction heuristic is described in Section 8.3. The important features of the tabu search algorithm with its embedded data structures are discussed in Section 8.4. Computational comparisons on a set of 20 test problems with the best published methods in the literature are provided in Section 8.5. Finally, our conclusion and perspectives on future research are presented in Section 8.6.

8.2 The Vehicle Fleet Mix (VFM) Problem

8.2.1 Representation of the VFM

Let '0' denote the depot with no service time and zero demand; the remaining notations are defined as follows:

K = The set of different vehicle types, $K = \{1, \dots, k_{max}\}$.

F_k = The vehicle fixed cost of type $k \in K$.

Q_k = The vehicle capacity of type $k \in K$.

T_k = The maximum travel time for the vehicle capacity of type $k \in K$.

v = The decision variable indicating the total number of mixed vehicles used in the final solution of the VFM problem.

V = The set of desired vehicles of different types, $V = \{1, \dots, v\}$, $V \subseteq K$.

N = The set of customers, $N = \{1, \dots, n\}$ where n is the total number of customers.

q_i = The demand of customer for $i \in N$,

δ_i = The service time of customer for $i \in N$.

d_{ij} = The distance between customers i and j, $d_{ij} = d_{ji}\, \forall i, j \in N \cup \{0\}$.

α_k = The variable cost per unit of distance for a vehicle of type k.

β_k = The time factor per unit of distance for a vehicle of type k.

R_p = The set of customers serviced by a vehicle $p \in V$.

σ = The function $\sigma : V \to K$, where $\sigma(p)$ indicates the smallest type of vehicles that can serve the customers in the set R_p.

π_p = The travelling salesman route which serves the set $R_p \cup \{0\}$, where $\pi_p(i)$ indicates the position of customer i in the route, π_p.

$D(\pi_p)$ = The total distance of the route, π_p.

$T(\pi_p)$ = The total travel time of the route, π_p.

$C(\pi_p)$ = The total variable and fixed cost of the route, π_p.

S= The feasible solution which is defined as $S = \{R_1, \ldots, R_v\}$.

Π = The set of all travelling salesman routes in S, $\Pi = \{\pi_1, \ldots, \pi_v\}$.

Our goal is to solve the following optimization problem:

$$\min_{S, \Pi, v} C(S) = \sum_{p \in V} C(\pi_p) \tag{8.1}$$

such that

$$\bigcup_{p \in V} R_p = N \text{ and } R_p \cap R_q = \emptyset \, \forall p \neq q \in V \tag{8.2}$$

$$\sum_{i \in R_p} q_i \leq Q_{\sigma(p)} \forall p \in V \tag{8.3}$$

$$D(\pi_p) = \sum_{i \in R_p \cup \{0\}} d_{i\pi_p(i)} \forall p \in V \tag{8.4}$$

$$T(\pi_p) = \beta_{\sigma(p)} \times D(\pi_p) + \sum_{i \in R_p} \delta_i \leq T_{\sigma(p)} \forall p \in V \tag{8.5}$$

$$C(\pi_p) = \alpha_{\sigma(p)} \times D(\pi_p) + F_{\sigma(p)} \forall p \in V \tag{8.6}$$

Constraint (8.2) ensures that each customer is supplied in one route. The set of constraints in (8.3) guarantees that the vehicle capacity is not exceeded. Equations in (8.4) represent the total sum of distance of each route, π_p. Since the TSP is a hard problem by itself, we have used approximate procedures to estimate each route, π_p. Equations (8.5) guarantee that the maximum travel time is not exceeded. Equations (8.6) represent the total cost per route including variable and fixed cost while Equation (8.1) is the total sum of costs in the solution to be minimized over all routes.

8.2.2 Literature Review of the VFM

There are few published works on the VFM problems. In the early 1980s, Golden, Assad, Levy and Gheysens[GALG84] were among the first to address this problem. They presented a mixed integer formulation, generated lower bounds on the optimal solution and described various heuristics based on the savings method of Clarke and Wright[CW64] and the route-first cluster-second algorithm. The later heuristic starts with a giant route that visits all customers but not the depot. This giant route is further improved by the Or-OPT exchange procedure before it is then partitioned into feasible routes. This heuristic is then repeated using a sample of *five* initial routes, hence leading to its name $(\text{MGT+OrOPT})^5$. Excellent results were reported using the $(\text{MGT+OrOPT})^5$ heuristic on problems ranging from 12 to 100 customers. Gheysens, Golden and Assad[GGA84] developed a cluster-first route-second heuristic. In the first stage, a heterogeneous fleet mix was obtained using a lower bound procedure developed in Golden, Assad, Levy and Gheysens[GALG84]. In the second stage, they adopted a generalized assignment based heuristic to solve the VRP problem using the vehicle fleet mix obtained in the first stage. This heuristic was also run for five different times using the fleet compositions associated with each of the five best lower bounds and it was denoted by LB5+VRP. Gheysens, Golden and Assad [GGA84] presented another mixed integer formulation for the VFM problem with time windows and performed a computational comparison of heuristics developed in Gheysens, Golden and Assad[GGA86]. Ferland and Michelon[FM88] showed that an exact method for the VRP with time windows and a homogeneous fleet size can be extended to the VRP with the heterogeneous fleet size. The proposed exact method was based on a column generation approach with relaxed time windows but no computational results were reported.

In the 1990s, Desrochers and Verhoog[DV91] presented a new savings heuristic called MBSA which was based on successive route fusions. At each iteration, the best fusion was selected by solving a weighted matching problem. The MBSA heuristic was implemented considering several weights in the matching problem such as the total savings in routing costs, savings in fixed costs, or opportunity savings associated with each feasible combination. Computational results were provided for a number of benchmark problems in order to compare the algorithm's performance to that of other methods. Ronen[Ron92] presented a mixed integer formulation to assign trips to mix fleet vehicles with the minimal cost. A two-step heuristic was proposed based on the assignment of trips to vehicles first, then slide-and-switch of trips between vehicles second. The heuristic provided results within 1% of the linear relaxation bound.

Salhi, Sari, Saidi and Touati[SSST92] presented a mixed integer formulation for the VFM problem with fixed and variable running costs. They assessed the effect of neglecting the variable running cost on the solution of different procedures. They modified the route-first cluster-second and the savings algorithms in order to take into account both variable running and fixed costs. This mixed integer formulation is similar to that of Gheysens, Golden and Assad[GGA84] which assumes the same value of the unit running cost across the different vehicle types. Salhi and Rand[SR93] presented a review on the VFM and also developed an interactive route perturbation procedure (RPERT) consisting of a series of refinement modules. New best-known solutions were reported by the RPERT procedure for some standard test problems. Finally, the importance of the strategic decisions on the fleet make-up and the vehicle number of each types was discussed in the context of distribution system design by Bookbinder and Reece[BR88] and Beaujon and Turnquist[BT91].

8.3 Modified RPERT Procedure (MRPERT)

Salhi and Rand[SR93] proposed a successful interactive procedure denoted by RPERT. The RPERT procedure constructs a VFM solution using a series of seven phases. Each phase uses one perturbation module in order to improve upon the solution of its predecessor. If the new solution is better and feasible in terms of vehicle capacities and maximum travel times, it is then accepted as the current solution and passed into the next phase for further improvements. Otherwise the previous solution is retained and the search continues until the RPERT procedure is terminated at the end of the last phase. It was noticed that in some modules, the routing cost was increased but with a greater decrease in the fixed cost. The aim was to explore the compromise between having a higher total operating cost and a lower total fixed cost, while achieving a maximum utilization of the whole fleet.

We shall briefly describe these modules in the same order as they were implemented in the RPERT procedure. For more details, the reader should refer to Salhi and Rand[SR93]. RPERT begins with the savings module developed for the vehicle routing problem in Salhi and Rand[SR87]. The savings module constructs an initial set of routes using a given vehicle capacity. This savings module uses a 'shape' parameter to modify the savings function used in [Daw86] to compute the savings in distance if pairs of customers were served in a single route. The shape parameter value was varied between 0 and 2 to generate a set of different initial routes. A *matching* module was then used to determine for each route the smallest type of vehicles that can serve the customers on it. This module was followed by a *reduction* module which was attempted, whenever possible, to eliminate a given route by relocating its customers and inserting them in other routes or merging the route with another to build a larger single one. A *sharing* module was also implemented. It attempted to split a given route into few smaller routes to achieve a cost reduction, if possible. Finally, the RPERT procedure improved the cost of routes by a *swapping* module which exchanged customers between routes. Let us define a *cycle of search* to be a single execution of the above sequence of the RPERT modules. The RPERT procedure was only performed using one cycle and was terminated at the end of its last module. The best stored solution at the end of this cycle was called the RPERT final solution.

In RPERT, a rigid restriction was imposed on the allowed type of *moves* in order to speed up the procedure. A move is a transition from one solution to another neighbouring solution. More precise, a move was not allowed to be performed if it resulted in a utilization of larger-sized vehicles than the currently used ones when implementing the reallocation and swapping modules, i.e, only *feasible* moves in terms of capacity and maximum travel were allowed and *infeasible* moves were prohibited. To alleviate these restrictions in this paper, we introduce the following two modifications. First, to enlarge the neighbourhood size (set of available moves), the feasibility restriction is relaxed, i.e., a move which leads to a utilization of a larger-sized vehicle was allowed. Second, we also allow the RPERT procedure to restart for another cycle starting from the best solution obtained at the end of previous cycle. The search continues for a number of cycles and it is terminated when a cycle is performed without finding any improvements. The reason is that RPERT performed only one cycle before terminating its search. However, if an improvement happened in any (say sharing) modules it may be possible to improve this solution by using an earlier (say reallocate or reduction) module. These further improvements would not be detected unless the RPERT procedure is restarted. The restart process and the allowance of infeasible moves were not implemented in RPERT. These modifications form the basis of the new procedure which is denoted by MRPERT.

8.4 Tabu Search

Tabu search (TS) is a relatively novel technique for solving hard combinatorial optimization problems. Tabu search ideas were proposed by Glover[Glo86]. It is based on selected concepts from artificial intelligence. Its goal is to emulate intelligent uses of memory for exploiting historical information. Tabu search uses memory structures to record in a *tabu list*, TABL, attributes of the recently accepted solutions, i.e. the changes occurred when performing a move from S to the best solution S' in its neighbourhood $N(S)$. Moreover, attributes can consist of customers removed, added or repositioned by the moves executed. Sometime, attributes can be strategically combined to create other attributes. Selected attributes, that are stored in TABL, are designated tabu-active. Solutions, that contain tabu-active elements, are designated *tabu.* The duration that an attribute remains on the tabu list and hence remains tabu-active, is called the *tabu-list size* which is denoted by t.

The approach of using attributes to identify the tabu status of future moves is very easy to implement and requires less storage than the approach of using the actual solutions which is very expensive to store and difficult to check and maintain. The attributes-based approach, however, may wrongly forbid moves leading to unvisited solutions that may be attractive. It is therefore necessary to override the tabu status of such moves, if an *aspiration level criterion* is satisfied, in order to correct this wrong diagnosis. A move is considered *admissible* if it is not a tabu move or its tabu status is overridden by the aspiration criterion.

In general, TS uses an aggressive guiding strategy to direct any local search procedure to carry out exploration of the solution space to avoid getting trapped in local optima. When a local optimum is encountered, the aggressive strategy moves to the best solution S' in the whole neighbourhood $N(S)$ even if it may cause a deterioration in the objective function value. For situations, where the neighbourhood is large or its elements are expensive to evaluate, *candidate list* strategies are used to help restrict the number of solutions examined on a given iteration. A *candidate list* of solutions $N'(S)$ is generated either randomly or strategically using memory structures[Osm93] in order to identify the best move exactly, or heuristically. Finally, the rule for execution is generally expressed by a *stopping* criterion. Several stopping criteria can be used to terminate the search: either a pre-specified limit on the number of iterations or on the number of iterations since the last improvement was found. Since TS embeds heuristic rules and different strategies to guide the search, it becomes a metastrategy algorithm or simply a metaheuristic. The basic TS procedure is sketched in Figure 8.1.

Tabu Search Procedure (TSVFM).

```
Step 1: initialization:
        - Generate an initial solution, S, for the VFM problem.
        - Set the best current solution $S_{best} = S$.
        - Evaluate all the moves in the neighbourhood $N(S)$.
        - Set values for: $t$, the tabu list size; TABL, the tabu list; DS,
        the special Data Structure for attributes;
        MAXBEST, the maximum number of iterations after $S_{best}$.
        - Set iteration counters: nbiter=0 (current iteration number) and
        bestiter= 0 (iteration number of the best solution).

Step 2: Candidate list of solutions:
        - Determine strategically using DS, the exact set of the candidate
        list of best moves in the neighbourhood, i.e., $N'(S) \subseteq N(S)$.
        - Update DS after each iteration.

Step 3: Selection Strategy:
        - Choose the best admissible solution $S' \in N'(S)$.
        - Set $S = S'$ and nbiter= nbiter +1.
        - If $C(S') < C(S_{best})$, then set $S_{best} = S'$ and bestiter = nbiter.
        - Record in TABL the changed attributes.

Step 4: Stopping criterion:
        If { (nbiter - bestiter) > MAXBEST }, then Stop,
        Else go to Step 1.
```

Figure 8.1 A basic TS procedure.

In the following, we shall give a description of the basic components of the above TS procedure and practical implementation details for solving the vehicle fleet mix problem. For recent details on tabu search, we refer to by Glover[Glo95a].

Initial solution: The initial solution, S, can be generated using any VFM or VRP heuristic. In this study, we have used the VRP heuristic of Salhi and Rand[SR87] to generate the initial set of routes, Π. The simple *matching* module of Salhi and Rand[SR93] is then used to determine for each route, π_p, the type of the vehicle, $\sigma(p)$, that can serve the set of customers R_p. The total cost C(S) of this VFM solution is computed using equations (8.1) and (8.6). An *empty* route with no customer assigned to it is added to the set of routes with zero costs. The reason for this empty route is to allow infeasible moves to be considered.

Neighbourhood generation mechanism: The neighbourhood generation mechanism describes how a given VFM solution $S = \{R_1, \ldots, R_p, \ldots, R_v\}$ can be altered to generate another solution S' in $N(S)$, the neighbourhood of S. Here, we adapt the λ-interchange mechanism which was introduced in Osman[Osm93] and successfully used for many problems, e.g., Chiang and Russell[CR96], Hasan and Osman[HO95], Osman[Osm95a], Thangiah, Osman and Sun[TOVS93, TOS94]. It has the property that λ-optimal solutions are μ-optimal solutions for any integer $\mu < \lambda$, Osman and Christofides[OC94].

Given a pair of route sets R_p and R_q in S, a 1-interchange generation mechanism invokes two processes to generate neighbouring solutions: a *shift process* which is represented by the (0,1), (1,0) operators and an *interchange process* which is represented by the (1,1) operator. Note that the reallocation and swapping procedures in Salhi and Rand[SR93] have some similarities with the shift and the interchange processes. The (1,0) shift process denotes the reallocation of one customer (say i) from the set of customers, R_p, to another set of customers, R_q. This shift process, if performed, would result in a new pairs of route sets: $R'_p = R_p - \{i\}$ and $R'_q = R_q \cup \{i\}$. The (0,1) shift process denotes a shift in the opposite direction, i.e., a shift of j from R_q to R_p and the new route sets become $R'_p = R_p \cup \{j\}$ and $R'_q = R_q - \{j\}$. The (1,1) interchange process combines the two shift processes simultaneously to generate compound moves to reach solutions that can not be generated by consecutively applying any of the shift process alone due to capacity/time restrictions. Each customer $i \in R_p$ is systematically attempted for interchange with every other customer $j \in R_q$ to get two new route sets $R'_p = (R_p - \{i\}) \cup \{j\}$ and $R'_q = (R_q - \{j\}) \cup \{i\}$.

The advantage of the (1,0) and (0,1) shift processes is that an empty vehicle may be produced or a single customer route may be formed. In both cases, savings in fixed costs may be obtained. To allow a single route to be formed using the shift operators, at least one empty set of customers, $R_\emptyset$, at any iteration needs to be available for usage among the other non-empty route sets. If a decrease in the vehicle number has occurred, we may have two empty route sets, one of which is redundant and can be deleted. Another advantage of the empty set is that it allows infeasible moves to be considered when searching the neighbours. Here, we have a non-monotonic increase/decrease in the number of vehicles as well as a non-monotonic search of feasible/infeasible regions. For more details on non-monotonic search strategies, we refer to Glover[Glo95b].

Finally, the neighbourhood, $N(S)$, is defined to be the set of all solutions that can be generated by considering a total number of $v(v+1)/2$ pairs which is equal to $v(v-1)/2 + v$ different pairs of route sets. The first term, $v(v-1)/2$, determines the number of pairs of route sets (R_p, R_q) where $1 \leq p < q \leq v$, involving v non-empty route sets, while the second term, v, is the number of pairs of route sets, $(R_p, R_\emptyset)\, \forall\, p \in V$, considered by the (1,0) process operating on the pair of non-empty and empty route sets. The customers in a given pairs of route sets are searched sequentially and systematically for improved solutions by the shift and interchange processes of the 1-interchange mechanism.

Evaluation of the Cost of a Move: Let 1IM denote a 1-interchange move from a current solution S to one of its neighbours $S' \in N(S)$ by either the shift or the interchange processes, i.e., 1IM $(S) = S'$. Given a pair of route sets R_p and R_q, the cost of the move, 1IM depends on the type of operators involved. There are three type of operators (1,0), (0,1) and (1,1).

(I) *The cost of (1,0)-based move.*
The (1,0)-based move involves shifting a customer i from the route π_p of R_p and inserting it in the best possible position on arcs (least cost insertion) of the approximate TSP route π_q of R_q. The (1,0) move would result in two route sets $R_{p'} = R_p - \{i\}$ and $R'_p = R_q \cup \{i\}$. Let a and b be the predecessor and the successor of i in the old TSP route π_p of R_p. Let us assume c and e to be the predecessor and the successor in the new TSP route π'_q of R'_q where i is to be best inserted. The costs of the new TSP routes are then computed as follows:

$$D(\pi'_p) = D(\pi_p) + d_{ab} - (d_{ai} + d_{ib}) \tag{8.7}$$
$$D(\pi'_q) = D(\pi_q) + d_{ci} - (d_{ie} + d_{ce}) \tag{8.8}$$
$$C(\pi'_p) = \alpha_{\sigma(p')} \times D(\pi'_p) + F_{\sigma(p')} \tag{8.9}$$
$$C(\pi'_q) = \alpha_{\sigma(q')} \times D(\pi'_q) + F_{\sigma(q')} \tag{8.10}$$
$$\Delta_{(1,0)} = (C(\pi'_p) - C(\pi_p)) + (C(\pi'_q) - C(\pi_q)) \tag{8.11}$$

where p' and q' are the vehicles that serve the new route sets R'_p and R'_q, respectively.

(II) *The cost of (0,1)-based move.*
The (0,1)-based move involves shifting a customer $j \in R_q$ to another route set R_p. This move is similar to the one in (I) except that R_p and R_q are in reversed order. Hence the cost of the (0,1) move is simply obtained by applying the (1,0) process on the route sets R_q, R_p in a similar way. Let (f, g) and (l, m) be the (predecessor, successor) of j in the routes π_p and π'_q respectively. Equations (8.7) to (8.11) can then be similarly rewritten as follows:

$$D(\pi'_p) = D(\pi_p) + d_{fj} - (d_{fj} + d_{fg}) \tag{8.12}$$
$$D(\pi'_q) = D(\pi_q) + d_{lm} - (d_{lj} + d_{jm}) \tag{8.13}$$
$$C(\pi'_p) = \alpha_{\sigma(p')} \times D(\pi'_p) + F_{\sigma(p')} \tag{8.14}$$
$$C(\pi'_q) = \alpha_{\sigma(q')} \times D(\pi'_q) + F_{\sigma(q')} \tag{8.15}$$
$$\Delta_{(1,0)} = (C(\pi'_p) - C(\pi_p)) + (C(\pi'_q) - C(\pi_q)) \tag{8.16}$$

where p' and q' are the vehicles that serve the new route sets R'_p and R'_q, respectively.

(III) *The cost of (1,1)-based move.*
The (1,1)-based move involves an exchange of $i \in R_p$ with $j \in$

R_q. This move can be seen as a combination of two simultaneous shifts: a shift of i into the route set ($R'_q = R_q - \{j\}$) by the (1,0) operator and another shift of j into the route set ($R'_p = R_p - \{i\}$) by the (0,1) operator. Note that, the change in the objective function value $\Delta_{(1,1)}$ due to the (1,1) operator can be easily evaluated from the stored data used for computing equations (8.7) to (8.11) and (8.12) to (8.16) bearing in mind some special cases that need careful examinations. For example, let us assume that the best insertion of customer i inside the route π'_q is recorded to be between two customers (c and e) in the old route π_q where j was included. If c and e are different from j, then the new insertion cost of i into π'_q is only the comparison between the old insertion cost of i in π_q with that along the newly created arc joining f to g (predecessor and successor of j) due to the removal of j. However, if either c or e is identical to j, then the insertion cost of i into π'_q needs to be evaluated in the same way as indicated in (I). Similar analysis needs to be conducted to find the insertion cost of j into π'_p. Then the computed costs will be used to evaluate the cost of the (1,1) move, $\Delta_{(1,1)}$.

The best 1-interchange move involving either i or j or both is determined from the values of the 1IM costs obtained by using the (1,0), (0,1) or (1,1) operators. Its value, δ_{ij}, is the most negative or the least positive change in costs and it is computed as:

$$\delta_{ij} = \min\{\Delta_{(1,0)}, \Delta_{(0,1)}, \Delta_{(1,1)}\} \tag{8.17}$$

The best 1-interchange move, $1\mathrm{IM}_{pq}$, involving the pair of routes (R_p, R_q) is the 1IM move which gives the best change in costs over all δ_{ij} obtained by searching systematically all possible shifts or exchanges of customers between R_p and R_q. Its value, Δ_{pq}, is computed as:

$$\Delta_{pq} = \min_{i \in R_p, j \in R_q} \delta_{ij} \tag{8.18}$$

The change in the objective function values, $\Delta = C(S') - C(S)$, is then set equal to Δ_{pq} if the best move, $1\mathrm{IM}_{pq}$, is performed using the set of routes (R_p, R_q). The $1\mathrm{IM}_{pq}$ move can be identified by its attributes: the move value, Δ_{pq}; the customer indices, i and j; the operator type; the route indices and the new/old predecessors and new/old successors. These attributes can be stored in a special data structure, DS, to be retrieved fast and save computational effort.

Data structures for the candidate list of moves: Data structures (DS) for strategically defining the set of best moves were introduced for the VRP in Osman[Osm93] and resulted in big savings of more than a half of the required computation time. Osman's DS stored only the set of best moves between all pairs of routes. However, it did not make use of any computed information generated in early evaluations in relation to the best insertion/removal positions and their associated costs. These attributes could be shared among many moves. Hence, storing them would generate further savings in computational effort. In this paper, we extend the previous data structure to include the above mentioned information. TS selects the best move, 1IM_{best}, from S to its best neighbours S' requiring the whole neighbourhood $N(S)$ to be re-evaluated after each iteration. The neighbourhood $N(S)$ consists of $v(v+1)/2$ pairs of routes which is very expensive to compute for large-sized problems. The set of best candidate moves $N'(S)$ in $N(S)$, which consists of the best 1IM_{pq} moves for all $1 \leq p < q \leq v$, can strategically be determined and updated using two levels of memory structures. The aim is to identify the changes and intelligently update the candidate list of 1IM_{pq} moves with the minimal effort without scarifying the quality of the 1IM_{best} move. At each iteration, we are able to find exactly the best move rather than an inferior one which can be obtained by random sampling of the neighbourhood. Hence, the importance of the DS structure can not be over-emphasized to achieve efficient TS implementations.

There are two levels of memory structures. The first level stores the costs of all 1-interchange moves, 1IM, in the neighbourhood of S. For instance, the cost $C(\pi'_p)$ of serving a route, π_p, without a customer, i, as evaluated in (8.7), can be stored in a cell COST_WITHOUT(p,i) of a $v \times n$ matrix. This value will remain unchanged once computed, during all the (1,0) and (0,1) processes of evaluating the cost of shifting customer i into all other different routes, for at least the current iteration. The COST_WITHOUT matrix can also help to evaluate quickly the cost of (1,1) move with a little extra effort as explained earlier. The memory concept is not only used for the move costs but can be extended to store other useful information on π'_p, such as its current load, its type, and its status in terms of capacity and time restrictions.

The second level of memory structure is mainly used to identify the set of all the best $v(v+1)/2$ moves, 1IM_{pq}, obtained when all $v(v+1)/2$ pairs of route sets, (R_p, R_q) for all $p < q \in V$, are considered by the 1-interchange mechanism to generate the neighbourhood $N(S)$ after each iteration. This set of best 1IM_{pq} moves defines the list of elite candidate solutions, $N'(S)$, in the neighbourhood. In Step 2 of the TS procedure, we select the best move, 1IM_{best}, from the set of best moves and apply it to the current solution S to obtain the next solution $S' \in N'(S)$. At this stage, S' is exactly the best solution in $N'(S)$ and in $N(S)$. Once, the best move, 1IM_{best}, is performed S' becomes the current solution for the next iteration. It can be seen after performing the best move that the change between S and S' is only in one pair of routes, say (R_p, R_q), and the other routes remain intact. Consequently, $2 \times (v-1)$ pairs of route sets, (R_p, R_m) and (R_m, R_q) (for all $m \in V$, $m \neq p$, $m \neq q$), are necessary to be re-evaluated in order to update the candidate list of best neighbouring solutions.

Having defined the set of elite moves and solutions, we shall describe our data structure (DS) to update the candidate list of moves and $N'(S)$ with a small number of re-evaluations. The data structure DS consists of two matrices. DSTABLE takes the form of a $v \times v$ matrix, whereas DSINF is a $\{v(v+1)/2\} \times A$ matrix where 'A' is the number of attributes required to identify a move in DS. Each entry of the top triangular part of DSTABLE is used to store the best change in the objective value associated with one of the $v(v+1)/2$ pairs of routes (R_p, R_q). For example, DSTABLE(p,q) stores Δ_{pq} associated with the best 1IM_{pq} between the pair (R_p, R_q), or a large positive value if no such move exists. The lower triangular part DSTABLE(q,p) is used to store a positional index l associated with the pair (R_p, R_q) in the set of possible pair combinations $\{1, ..., v(v+1)/2\}$. Note that the determination of the index needs special care as the number of vehicles may increase during the computation. Increasing the vehicle number option was not considered in Osman's data structure whereas it is of a primary importance in the VFM.

In this study, we have used a value of six for 'A' indicating that six attributes are to be saved in DSINF. For instance, the best move, 1IM_{pq}, between R_p and R_q is identified by its Δ_{pq} value in DSTABLE, the index l associated with the (R_p, R_q) pair that is used to identify other attributes in DSINF as follows: $\text{DSINF}(l,1) = l$, $\text{DSINF}(l,2) = p$, $\text{DSINF}(l,3) = q$, $\text{DSINF}(l,4) = i$, $\text{DSINF}(l,5) = j$, $\text{DSINF}(l,6) = \Delta_{pq}$. Note that both $\text{DSINF}(l,1)$ and $\text{DSINF}(l,6)$ could be ignored as they are already recorded in DSTABLE, but are introduced for convenience. If Δ_{pq} is associated with (1,0) or (0,1) operators, then we replace the values of i or j by $\emptyset$ in DSINF so that we can identify easily the type of the operators applied to this move. The columns in DSINF can be increased easily to store more information, if necessary.

The management of DS occurs in Step 1 of the TS procedure. At the first iteration, all the moves in the neighbourhood N(S) are evaluated and the best moves are stored in the appropriate matrices. When the TS procedure returns to this step, the DSTABLE matrix is updated considering only the $2 \times (v-1)$ pairs of routes that may contain improved moves due to performing the previous move. After this quick update, DSTABLE is scanned to identify the best move, 1IM_{best}, in $N'(S)$ for the next iteration. 1IM_{best} is identified by comparing the Δ_{pq} stored values of the 1IM_{pq} moves. The attributes of the identified move are then retrieved from DSINF. The corresponding new routes, π'_p and π'_q, are further checked for possible improvements applying the 2-OPT or 3-OPT post optimization procedures of Lin[Lin65]. After the post optimization procedure is applied, this best move, 1IM_{best}, is performed to update the current solution. The tabu list is also updated before the search continues for another iteration. The selection of the best moves based on the above data structure has an advantage over the random sampling approach as it is performed with the minimal computational effort. Random sampling is normally used to reduce the computational effort as the systematic search is expensive to use without a data structure. Consequently, good moves may be missed and can not be detected unless a systematic search is used.

Tabu list: TABL takes the form of a $(v+1) \times n$ matrix (v rows: one for each route set R_p, one for the empty route; n columns: one per customer). In the tabu list, we store some attributes of the best 1-interchange move, 1IM_{best} which was accepted to transform S to $S' \in N(S)$. To prevent reversing the accepted move from returning back to S, its attributes are stored in TABL for t iterations, in the hope that we would be in a different region. The reverse move can be regenerated by returning immediately i to R_p and j to R_q. Let us assume that the current iteration number is *nbiter* and the best 1-interchange move involves the pair of customers (i, j) in (R_p, R_q). The return to previous solutions during the next t iterations can be prevented by storing in TABL the future value $(nbiter + t)$ in both $\text{TABL}(i, p)$ and $\text{TABL}(j, q)$ entries. After *nbiter+t* iterations, customers i and j are allowed to return to R_p and R_q, respectively.

Tabu condition: One way to prevent cycling is to store exactly all previously accepted solutions, but this approach can be computationally very expensive for checking and maintaining solutions. Another approach is to use hashing functions (Hasan and Osman[HO95]) to represent these solutions. We found in many implementations that tabu conditions based on move attributes provided good results. Therefore, we did not try other types of tabu conditions and the reader may refer to Osman[Osm95a] for a comprehensive computational study on the effect of different types of tabu conditions on the quality of solution.

In this paper, we implemented one type of tabu condition. A 1-interchange move, 1IM, is considered tabu if i is returned to R_p and j is returned to R_q. At iteration *nbiter*, we can check the status of a move 1IM involving the pairs (i, j) and (R_p, R_q) using the simple test in equation (8.19) and the information in TABL as follows. A move is considered tabu if

$$\text{TABL}\,(p, i) > nbiter \text{ and } \text{TABL}(q, j) > nbiter$$

TABL is initialized with zeros so that all moves are not considered tabu by the test.

Tabu list size: Tabu list size seems to be related to tabu conditions, selection strategy of neighbours and problem characteristics, such as the number of customers, the number of vehicles and the ratio of the total required demands to the available capacities. Osman[Osm93] established such linkages and used a statistically derived formula to find an estimate of the tabu list size, t. Then, in a systematic way, the tabu list sizes were varied every $2 \times t$ iterations over three different values ($\pm 10\%$ centred at the estimate). In this paper, a simpler way is used with a static value of t such as $\left\lceil \frac{n}{p} \right\rceil$ for p varying between 2 and 7 to watch the best range between these extremes. The aim is to identify the value of t for other problems. A small value may lead to cycling with poor results while a large value may lead to deterioration in solution quality caused by forbidding too many moves.

Aspiration criterion: Tabu conditions are based on stored attributes of moves; it may be possible to forbid wrongly unvisited solutions which share these attributes. Aspiration criteria are tests to correct such prevention. We use the following aspiration criterion. If 1IM is a tabu move but gives a new solution which is better than the best found so far, then we drop its tabu status. Therefore, a move from S to S' is considered admissible, if it is a non-tabu move, or a tabu move which passed an aspiration level criterion, $C(S') < C(S_{best})$.

Stopping criterion: Our TS procedure is terminated after a pre-specified number of iterations MAXBEST$= 5 \times n$ is performed after the best iteration number, bestiter, at which the best solution was found without finding any improvement.

8.5 Computational Experience

It is a common practice to compare the performance of a heuristic with that of existing techniques on a set of *benchmarks* (test problems) in terms of solution quality and computer requirements. Other measures such as simplicity, flexibility, ease of control, interaction and friendliness are also of interest but are not considered here.

Test problems: A set of 20 test problems are used. These problems were proposed by Golden, Assad, Levy and Gheysens[HN93]. They vary in size from 12 to 100 customers, there is no restrictions on maximum travel time ($T_k = \infty$, for all $k \in K$) nor on the number of vehicles. There are at least three types of vehicles and can be up to six for some problem instances, i.e., k varies between 3 and 6. The unit running cost, α_k, and β_k are set to a constant equal to one ($\alpha_k = 1, \beta_k = 1$, for all $k \in K$) to be able to compare with published results which do not consider these two factors. However, the fixed cost, F_k, increases with the vehicle capacities.

Quality of solutions: A common measure to decide on the quality of a heuristic solution is the relative percentage deviation from the optimal solution. However, if an optimal solution can always be easily obtained, we do not need to use a heuristic. Due to the difficulty of finding an optimal value, either a good lower bound from a linear relaxation of a mixed integer formulation or the best known solution can be used to replace the optimal value in the analysis. For each test problem, the relative percentage deviation over the best known solution is computed as follows:

$$\left\{ \frac{\text{heuristic solution - best known solution}}{\text{best known solution}} \right\} \times 100$$

In this study, we also report the actual solution, the average relative percentage and standard deviations, and the number of times a particular heuristic found the best solution. In Table 8.1, the actual solutions for our heuristics are given. The first and second columns give the problem number and the number of customers, respectively. The results of RPERT are in Salhi and Rand[SR93], our results of the MRPERT (modified RPERT) and the TSVFM (tabu search) with their respective CPU computation time in seconds on VAX 4500 are next in the table. The 'Old Best' are the cost of the best known solutions in the literature. These results can be found in either Golden, Assad, Levy and Gheysens[GALG84] or Gheysens, Golden and Assad[GGA84], Desrochers and Verhoog[DV91], and Salhi and Rand[SR93]. The final column contains the 'New Best' known solutions adjusted to reflect our findings. The new best solutions are indicated in bold while the references of the old best solutions are put between brackets [.]. It can be seen from Table 8.1 that we have found seventeen new or equal best known solutions for the twenty test problems using either the TSVFM or MRPERT heuristics. In fact, twelve out of sixteen are new best known solutions and the other five problems are equal to the published ones. Looking at the performance of each individual heuristic, we notice that the tabu search heuristic, TSVFM, has obtained fifteen best known values out of which ten are new best results. The modified heuristic, MRPERT, has obtained six best known solutions out of which two are new best results. MRPERT provides an improvement over RPERT in thirteen problems and is equal in the remaining instances.

Table 8.1 Comparison of the best known solutions

No.	Size	RPERT	MRPERT	CPU	TSVFM	CPU	Old Best	New Best
1	12	614	606	0.3	**602**	3	**602**[TL92]	**602** [*]
2	12	**722**	**722**	0.2	**722**	2	**722**[GALG84]	**722** [*]
3	20	1003	971.95	0.7	971.24	5	**965**[HO95]	965
4	20	6447	6447.80	0.2	**6445.10**	6	6446[GALG84]	6445 [*]
5	20	1015	1015.13	0.2	**1009.15**	5	1013[GALG84]	1009 [*]
6	20	**6516**	**6516.56**	0.2	**6516.56**	4	**6516**[SR93]	**6516** [*]
7	30	7402	7377	4.6	7310	15	**7298**[GALG84]	7298
8	30	2367	2352	3.4	**2348**	17	2349[GALG84]	**2348** [*]
9	30	**2209**	**2209**	0.8	**2209**	14	**2209**[SR93]	**2209** [*]
10	30	2377	2377	0.6	**2363**	14	2368[GALG84]	**2363** [*]
11	30	4819	4787	0.4	**4755**	19	4763[GALG84]	**4755** [*]
12	30	**4092**	**4092**	0.3	**4092**	10	**4092**[SR93]	**4092** [*]
13	50	2493	2462.01	7.8	2471.07	62	**2437**[GALG84]	2437
14	50	9153	9141.69	9.3	**9125.65**	71	9132[GALG84]	**9125** [*]
15	50	2623	**2600.31**	2.8	2606.72	46	2621[TL92]	**2600** [*]
16	50	2765	2745.04	1.2	**2745.01**	35	2765[SR93]	**2745** [*]
17	75	1767	1766.81	6.3	**1762.05**	85	1767[SR93]	**1762** [*]
18	75	2439	2439.40	4.5	**2412.56**	116	2432[GALG84]	**2412** [*]
19	100	8751	8704.20	8.1	**8685.71**	289	8700[TL92]	**8685** [*]
20	100	4187	**4166.03**	61.1	4188.73	306	4187[SR93]	**4166** [*]

[x]:	**x** indicates the reference in which the best known solution is reported and, * refers to this paper.
RPERT:	Interactive procedure of Salhi and Rand[SR93].
MRPERT:	The modified RPERT in this paper.
TSVFM:	The Tabu search procedure.
CPU:	The CPU time in seconds on VAX 4500.

Table 8.2 The relative percentage deviation above the best known solution.

No	Size	(MGT-OrOPT)^5	LB5+ VRP	MBSA	RPERT	MRPERT	TSVFM	OLBBEST
1	12	3.32	**2.65**	0.00	1.99	0.66	0.00	0.00
2	12	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	20	0.10	0.31	0.10	**3.93**	0.72	0.62	0.00
4	20	**7.52**	0.09	1.70	0.03	0.03	0.00	0.01
5	20	0.39	2.08	2.27	0.59	0.59	0.00	0.39
6	20	7.02	0.03	0.01	0.00	0.00	0.00	0.00
7	30	1.24	0.76	1.69	1.42	1.08	0.16	0.00
8	30	0.80	0.59	0.21	0.80	0.17	0.00	0.04
9	30	0.49	2.39	1.26	0.00	0.00	0.00	0.00
10	30	0.29	1.05	2.28	0.59	0.59	0.00	0.21
11	30	0.16	0.69	2.56	1.34	0.67	0.00	0.16
12	30	1.07	0.97	3.39	0.00	0.00	0.00	0.00
13	50	0.04	-	0.28	2.29	1.02	**1.39**	0.00
14	50	0.06	0.32	0.06	0.29	0.17	0.00	0.06
15	50	1.53	0.80	0.80	0.88	0.00	0.23	0.08
16	50	2.80	-	2.33	0.72	0.14	0.00	0.72
17	75	1.19	-	**6.52**	0.28	0.28	0.00	0.28
18	75	0.82	-	3.10	1.11	**1.11**	0.00	0.82
19	100	0.41	-	0.17	0.75	0.21	0.00	0.17
20	100	0.69	-	1.99	0.50	0.00	0.55	0.50
Average deviation		1.50	0.91	1.54	0.88	0.36	**0.14**	0.21
Standard deviation		2.10	0.83	1.59	0.94	0.39	0.35	0.28
# best solutions		1	1	2	4	7	**14**	8

(MGT-OrOPT)5:	Route first-cluster second algorithm of Golden et al.[GALG84].
LB5+VRP:	Cluster First-route Second algorithm of Gheysens et al.[GGA84].
MBSA:	Matching based savings algorithms of Desrochers and Verhoog[DV91].
Bold:	shows the worst instance performance of each algorithm and the best average over all problems.
Others:	As explained in Table 8.1.

In Table 8.2, we have reported the relative percentage deviation over the new best known solutions for most of the state-of-the-art algorithms in the literature. We have included those heuristics which have produced at least one of the best known solutions. The first row contains the headers of the considered heuristics: (MGT-OrOPT)[5] is the route first-cluster second heuristic based on route partitioning approach by Golden, Assad, Levy and Gheysens[GALG84]; LB5+VRP is the cluster first-route second heuristic based on the generalized assignment approach by Gheysens, Golden and Assad[GGA84]; MBSA is the matching based savings heuristic by Desrochers and Verhoog[DV91] and those remaining are our own algorithms. It should be noted that the relative percentage deviations for the MBSA column are the best relative percentage deviations from several variants (CM, OOM, ROM, ROM-γ, ROM-ρ) of the matching based savings heuristic. Desrochers and Verhoog[TL92] did not report the objective solution values, we had to estimate these results from their best percentage deviations in order to compute the data in the MBSA column.

It can be seen from Table 8.2 that TSVFM has the best performance as it produces the largest number of best known solutions and the smallest average relative percentage deviation of 0.14%. The MRPERT, however, produces the second largest number of best solutions, has the second average percentage deviation and the least worst percentage deviation value. These best average and worst relative percentage deviations for each heuristic are shown in bold in Table 8.2. The statistical analysis demonstrates the robustness of our heuristics when compared to the best available algorithms in the literature.

Computational effort. The computer programs are coded in Fortran 77 and run on a VAX 4500 computer. In the previous section, we have shown that TSVFM is the best heuristic; however, there is a cost for a such good performance. In Table 8.1, the CPU time for both TSVFM and MRPERT heuristics are reported and the average CPU time over all the twenty instances for TSVFM and MRPERT are 56.2 and 5.7 CPU seconds respectively.

Comments: In this section, we discuss observations and issues related to our experimental results. In Table 8.1, we observe that MRPERT has produced better solutions than TSVFM in three problems, namely: 13, 15 and 20. We attempted to see if any link may exist between this failure and the existence of many vehicle types since problem 13 has six different vehicle types. We are not yet able to justify the link since TSVFM has produced results better than or equal to MRPERT for other problems of similar types. Further, both methods have failed to find the best known solutions for three problems. The failure may be attributed to the combinatorial structure of these problems. In our view this observation may need a closer look. A further study is needed to find a concrete answer for this failure.

The second observation is that there is a good range for t values which vary with the size of problems. Figure 8.2 shows the average relative and standard percentage deviations of the best solutions for different values for the tabu list sizes, $t = \left\lceil \frac{n}{p} \right\rceil$ for $p = 2$ to 7, from the best known solutions for all the twenty test problems. From the figure, it can be seen that the performance of tabu search depends on the values of t. A good range for t values seem to lie between $\lceil \frac{n}{5} \rceil$ and $\lceil \frac{n}{3} \rceil$. Outside this range t is either too small or too big. In the former case, cycling would occur and relatively bad solutions would be generated. In the later case, many moves are prevented leading to unvisited solutions that may be attractive and longer computation time is needed to find good solutions. It was found that when using the right interval of the tabu list size value, the total CPU time required by the TS algorithm was reduced by a half of that needed for other values of p. To derive a better range for tabu list size, we could generate a random permutation of $\left\lceil \frac{n}{p} \right\rceil$ values with $p = 3$, 4 and 5 and let t takes each value in this permutation. Then t can be kept at each value, say T, for $2 \times T$ iterations before it can be changed to take the other values. Varying tabu list size values has been shown to be effective in Taillard[Tai93] and Osman[Osm93]. Other recent approaches based on a reverse elimination method or the reactive tabu method can be used and merit further investigations. These approaches are reviewed in Battiti[Bat96] and Voß[Vos96].

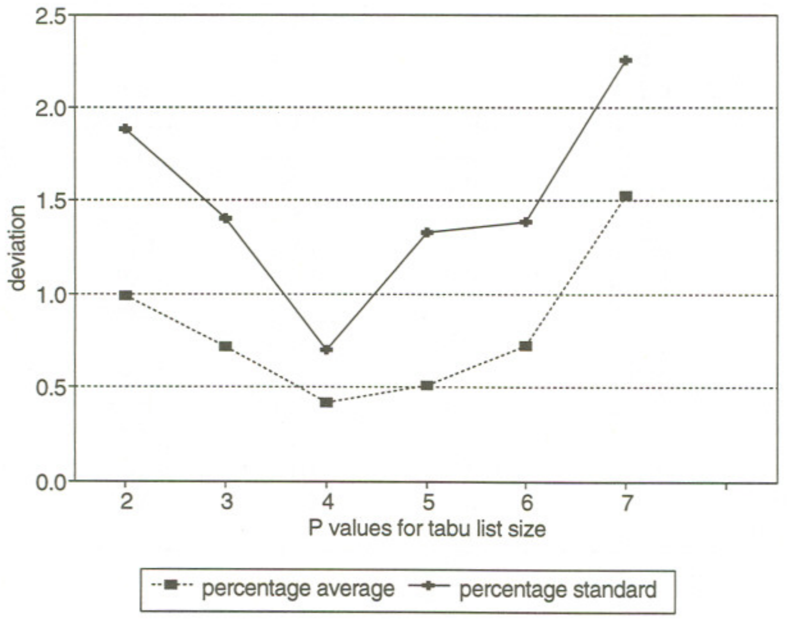

Figure 8.2 Effect of tabu list size on the quality of solution.

The initial starting solution was generated by solving a VRP problem using one vehicle type at a time by Salhi and Rand VRP's heuristic. The best VRP solution is then used to start both MRPERT and TSVFM for further improvements. No other starting solutions were considered for convenience, simplicity and consistency. It may be possible to use other VFM heuristics and lower bounds on the fleet composition

REFERENCES

[AL96] Aarts E. H. L. and Lenstra J. K. (1996) *Local Search in Combinatorial Optimization*. John Wiley, Chichester.

[Bat96] Battiti R. (1996) Towards self-tuning heuristics. In Rayward-Smith V. J., Osman I. H., Reeves C. R., and Smith G. D. (eds) *Modern Heuristic Search Methods*. John Wiley, Chichester.

[BR88] Bookbinder J. H. and Reece K. E. (1988) Vehicle routing considerations in distribution and system design. *European Journal of Operational Research* 37: 204–213.

[BT91] Beaujon G. L. and Turnquist M. A. (1991) A model for the fleet sizing and vehicle allocation. *Transportation Science* 25: 19–45.

[CR96] Chiang W.-C. and Russell R. (1996) Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research* 63. Baltzer, Basel.

[CW64] Clarke G. and Wright J. W. (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12: 568–581.

[DV91] Deroschers M. and Verhoog T. W. (1991) A new heuristic for the fleet size and mix vehicle routing problem. *Computers and Operations Research* 18: 263–274.

[EL96] Eglese R. W. and Li L. (1996) Heuristics for arc routing with a capacity constraint and time deadline. In Osman I. H. and Kelly J. P. (eds) *Metaheuristics, Theory and Applications*. Kluwer, Boston.

[FM88] Ferland J. A. and Michelon P. (1988) The vehicle routing with multiple types. *Journal of Operational Research Society* 39: 577–583.

[GALG84] Golden B. L., Assad A., Levy L., and Gheysens F. G. (1984) The fleet size and mix vehicle routing problem. *Computers and Operational Research* 11: 49–66.

[GGA84] Gheysens F., Golden B. L., and Assad A. (1984) A comparison of techniques for solving the fleet size and mix vehicle routing problem. *OR Spektrum* 6: 207–216.

[GGA86] Gheysens F. G., Golden B. L., and Assad A. (1986) A new heuristic for determining fleet size and composition. *Mathematical Programming Studies* 26: 233–236.

[GHL94] Gendreau M., Hertz A. and Laporte G. (1994) A tabu search heuristic for the vehicle routing problem. *Management Science* 40: 1276–1290.

[Glo86] Glover F. (1986) Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research* 13: 533–549.

[Glo95a] Glover F. (1995) Tabu search fundamentals and uses. Technical report, University of Colorado Boulder. Working Paper.

[Glo95b] Glover F. (1995) Tabu thresholding: Improved search by non-monotonic trajectories. *ORSA Journal on Computing* 7: 426–442.

[HO95] Hasan M. and Osman I. H. (1995) Local search strategies for the maximal planar graph. *International Transactions in Operational Research* 2: 89–106.

[Lin65] Lin S. (1965) Computer solutions to the travelling salesman problem. *Bell System Technology Journal* 44: 2245–2269.

[LO95] Laporte G. and Osman I. H. (1995) Routing problems: A bibliography. *Annals of Operations Research* 61: 227–262.

[LO96] Laporte G. and Osman I. H. (1996) Metaheuristics in combinatorial optimization. *Annals of Operations Research* 63. Baltzer, Basel.

[LR81] Lenstra J. K. and Rinnooy Kan A. H. G. (1981) Complexity of vehicle routing and scheduling problem. *Networks* 11: 221–227.

[OC94] Osman I. H. and Christofides N. (1994) Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research* 1: 317–336.

[OK96a] Osman I. H. and Kelly J. P. (1996) *Metaheuristics. Theory and*

Applications. Kluwer, Boston.

[OK96b] Osman I. and Kelly J. (1996) Metaheuristics. an overview. In Osman I. H. and Kelly J. P. (eds) *Metaheuristics, Theory and Applications.* Kluwer, Boston.

[OL96] Osman I. H. and Laporte G. (1996) Metaheuristics: A bibliography. *Annals of Operations Research on Metaheuristics* 63. Baltzer, Basel.

[Osm93] Osman I. H. (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operational Research* 41: 421–451.

[Osm95a] Osman I. H. (1995) Heuristics for the generalised assignment problem. simulated annealing and tabu search approaches. *OR Spektrum* 17: 211–225.

[Osm95b] Osman I. H. (1995) An introduction to metaheuristics. In Lawrence M. and Wilsdon C. (eds) *Operational Research Tutorial Papers.* Operational Research Society, Birmingham.

[PKGR96] Potvin J.-Y., Kervahut T., Garcia B., and Rosseau J. (1996) A tabu search heuristic for the vehicle routing problem with time windows. *ORSA Journal on Computing* ,Forthcoming.

[Ree93] Reeves C. R. (1993) *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell, Oxford.

[Ron92] Ronen D. (1992) Allocation of trips to trucks operating from a single terminal. *Computers and Operations Research* 19: 451–451.

[RR96] Rego C. and Roucairol C. (1996) A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In Osman I. H. and Kelly J. P. (eds) *Metaheuristics, Theory and Applications.* Kluwer, Boston.

[RS95] Rayward-Smith V. J. (1995) *Applications of Modern Heuristic Methods.* Alfred Waller, Oxford.

[SR87] Salhi S. and Rand G. K. (1987) Improvements to vehicle routing heuristics. *Journal of Operational Research Society* 38: 293–295.

[SR93] Salhi S. and Rand G. K. (1993) Incorporating vehicle routing into the vehicle fleet composition problem. *European Journal of Operational Research* 66: 313–330.

[SS95] Salhi S. and Sari M. (1995) A heuristic approach for the multi-depot vehicle fleet mix problem. Technical report, School of Mathematics and Statistics, University of Birmingham, U.K.

[SSST92] Salhi S., Sari M., Saidi D., and Touati N. A. C. (1992) Adaptation of some vehicle fleet mix heuristics. *OMEGA* 20: 653–660.

[Tai93] Taillard E. D. (1993) Parallel iterative search methods for vehicle routing problems. *Networks* 23: 661–673.

[TOS94] Thangiah S., Osman I., and Sun T. (1994) Metaheuristics for the vehicle routing problems with time windows. Technical Report UKC/IMS/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury.

[TOVS93] Thangiah S., Osman I., Vinayagamoorthy R., and Sun T. (1993) Algorithms for vehicle routing problems with time deadlines. *American Journal of Mathematical and Management Sciences* 13: 323–357.

[Vos96] Voss S. (1996) Observing logical interdependencies in tabu search. methods and results. In Rayward-Smith V. J., Osman I. H., Reeves C. R., and Smith G. D. (eds) *Modern Heuristic Search Methods.* John Wiley, Chichester.