The final version of this article has been published in:
Brandão, J. (2011). A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. Computers & Operations Research, 38 (1), 140-151. doi: 10.1016/j.cor.2010.04.008.

A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem

José Brandão

Departamento de Gestão, Escola de Economia e Gestão, Universidade do Minho, Largo do Paço, 4704 – 553, Braga, Portugal.

CEMAPRE, ISEG, Technical University of Lisbon, Portugal.

E-mail: sbrandao@eeg.uminho.pt

Abstract

In the heterogeneous fixed fleet vehicle routing problem there are different types of vehicles and a given number of vehicles of each type. The resolution of this problem consists of assigning the customers to the existing vehicles and, in relation to each vehicle, defining the order of visiting each customer for the delivery or collection of goods. The objective is to minimize the total costs, satisfying customers' requirements and visiting each customer exactly once. In this paper a tabu search algorithm is proposed and tested on several benchmark problems. The computational experiments show that the proposed algorithm produces high quality solutions within an acceptable computation time. Four new best solutions are reported for a set of test problems used in the literature.

Keywords: Heterogeneous fixed fleet; Vehicle routing; Tabu search; Heuristics; Logistics

1. Introduction

The *heterogeneous fixed fleet vehicle routing problem* (HFFVRP) is defined on an undirected connected graph $G = (V, E)$, where $V = \{0, 1, \ldots, n\}$ is a vertex set and $E = \{(i, j): i, j \in V\}$ is an edge set. Vertex *0* represents the depot and each other vertex $i \in V \setminus \{0\}$ is a customer with a demand q_i. A distance d_{ij} ($d_{ii} = 0$, $\forall i \in V$) is associated to each edge $(i, j) \in E$. There is a fleet of T different types of vehicles located at the depot, and the number of vehicles of each type is n_k ($k = 1, \ldots, T$). A capacity Q_k and a variable cost v_k are associated to each type of vehicle k ($k = 1, \ldots, T$). We assume that $Q_1 < Q_2 \ldots < Q_T$ and $v_1 < v_2 \ldots < v_T$. The travelling cost of each edge $(i, j) \in E$ by a vehicle of type k is $c_{ij} = v_k d_{ij}$. The HFFVRP consists of defining a set of routes and the vehicles assigned to them so that the following constraints are taken into account: *i)* use no more vehicles than those available; *ii)* satisfy customers' demand; *iii)* visit each customer exactly once; *iv)* a vehicle route starts and finishes at the depot; *v)* do not exceed the

capacity of the vehicle. We will refer indistinctly to route type or vehicle type because each route is assigned to one vehicle. The objective of the HFFVRP is to minimize the sum of the costs of all the routes subject to the previous constraints. Considering the fact that the number of vehicles of each type is fixed, there is no guarantee that a feasible solution can be found. This means that, if this happens in real-life problems, other vehicles have to be hired in order to serve the customers.

The HFFVRP is an NP-*hard* combinatorial problem, since it reduces to the *capacitated vehicle routing problem* (CVRP) if the number of types of vehicles is just one, i.e., the vehicle fleet is homogeneous, and the number of vehicles is unlimited.

If we assume that the number of vehicles of each type is unlimited, we get another type of problem known as the *fleet size and mix vehicle routing problem* (FSMVRP). Although very similar, these two types of problems are applied in rather different situations: the FSMVRP is more appropriate for strategic decisions when a company wants to buy a vehicle fleet and needs to define its size and composition, while the HFFVRP represents better the operational decisions of defining the vehicles that should be used, among those available, in order to serve the customers. On the other hand, the HFFVRP has been much less studied than the FSMVRP, and is rather harder to solve, according to our experience. This additional difficulty might be the reason why, as conjectured by Taillard [1], it has received less attention from researchers. When the ratio between the total demand of the customers and the total capacity of the vehicles is close to one even finding a feasible solution can be very difficult. In the general case, to prove if a given homogeneous fleet is enough to satisfy the demand requires to solve a bin packing problem, which is an NP-hard problem (Garey and Johnson [2]). If the fleet is heterogeneous, we deal with an extension of the bin packing problem with bins of different sizes and, therefore, it is also NP-hard, but much less studied than the problem with only one type of bin (see, for example, Correia et al. [3]) and much more difficult to solve.

In the literature there are many variants of the heterogeneous vehicle routing problem, besides the two mentioned before. The interested reader can find a comprehensive classification in Paraskevopoulos et al. [4], jointly with several representative articles. In this article the authors present an algorithm to solve both the FSMVRP and the HFFVRP with time windows. In the case of the FSMVRP, several references and applications can also be found in Brandão [5].

In practice, the need for different types of vehicles is determined by the characteristics of the customers. Usually, larger vehicles are more appropriate for

serving customers who require large orders (because this avoids splitting the order and reduces the total distance travelled by serving several customers in the same route) and the smaller vehicles are more adequate to deliver small quantities or serve customers that have access restrictions. For the daily planning the most important costs to be considered in the objective function are the variable costs. Therefore, we omitted the fixed costs as is the case with all researchers that have investigated the HFFVRP. In Li et al. [6] can be found several examples of practical applications of the HFFVRP. Some authors, as for example, Semet and Taillard [7], Rochat and Semet [8], Brandão and Mercer [9] and Oppen and Lokketangen [10], have studied real-life problems that can be classified as extensions of the HFFVRP, since they comprise several additional constraints, like time windows at the customers for the deliveries, maximum driving time, etc. Below, we will briefly review some of the literature on HFFVRP resolution methods.

As far as we know, the only papers published about the HFFVRP are the following: Taillard [1], Tarantilis et al. [11, 12] and Li et al. [6].

The method proposed by Taillard [1] takes advantage of the similarities between the VRP and the HFFVRP. The author solves a VRP, for each type of vehicle, using the adaptive memory procedure developed by Rochat and Taillard [13], generating a large number of routes. Then, in order to find from these routes the best solution for the HFFVRP, the author uses a linear programming algorithm called column generation.

Tarantilis et al. [11, 12] used a stochastic search method, belonging to the class of threshold accepting based methods (Dueck and Scheuer [14]), where the threshold parameter plays a major role in guiding the search process. In [11] the authors use a list of threshold values in the search, while in [12] the main feature is that, during the search, the threshold value can suffer an occasional increase instead of decreasing monotonically as is typical of this kind of methods. These two algorithms are called, respectively, list based threshold accepting (LBTA) and backtracking adaptive threshold accepting (BATA). The kinds of moves used in both algorithms are 2-opt, 1-1 exchange and 1-0 exchange. All these types of moves are applied inside each route and among all the routes of the current solution. The type of move performed in each iteration and the customers used in the move are chosen randomly.

Li et al. [6] adapted their record-to-record travel algorithm for the VRP [15] to solve the HFFVRP, which was named HRTR. The methodology used by them is based on Dueck's [16] record-to-record travel, which is a deterministic variant of simulated

annealing. In their algorithm are used the following types of moves: one point (node), two point, 2-opt and or-opt, all of which can be performed within and between routes. If after a given number of iterations no new better solution is found the best known solution is perturbed, and the process restarts with this solution.

Since we have created an algorithm for the FSMVRP (Brandão [5]), our first attempt was to adapt it, making a few changes, to solve the HFFVRP. However, the experiments have demonstrated that these changes were not enough to find competitive solutions comparing to the other algorithms published. Therefore, although several characteristics of that algorithm were maintained some significant changes had to be introduced.

The remainder of this paper is organized as follows. Section 2 describes the tabu search algorithm (TSA) for solving the HFFVRP, presenting, firstly, its main features and then a step by step description that shows how the different components work together and, finally, the proposed parameters setting is discussed. In Section 3 are given the results obtained on three sets of test problems, and the final conclusions are presented in Section 4.

2. The tabu search algorithm

The tabu search (TS) is a metaheuristic due to Glover [18, 19, 20] that was inspired by the principles of Artificial Intelligence, especially, the use of memory. The tabu list is one of the ways of using memory, by storing in a list the solutions explored throughout the search or, more commonly, some relevant attributes of these solutions. This tabu list has two main purposes: to prevent the return to the most recent visited solutions in order to avoid cycling; to drive the search towards regions of the solution space not yet explored and with high potential of containing good solutions.

A successful application of TS requires a good balance between intensification and diversification. The intensification is a detailed exploration of some region of the solution space, usually in the vicinity of a good solution. The diversification consists in driving the search towards promising regions of the solution space not yet explored and apart from one another, in terms of structure.

In operational terms, the TS consists of moving successively, in each iteration, from one solution S to the best of its neighbour solutions, $N(S)$, not in the tabu list or, if so, satisfying some aspiration criterion.

Our TSA uses and explores most of the more important concepts in TS, namely, the following: attributes, tabu list, tabu tenure, aspiration criteria, intensification, diversification, frequency-based memory, strategic oscillation, neighbourhood and neighbourhood size, moves and evaluation of the moves.

2.1. Method for generating the initial solution – giant tour

The execution of the TSA starts with an initial solution. Usually the methods used to generate an initial solution are simple and fast to compute because it is assumed that the task of producing a good solution is mainly committed to the TSA. As explained in the previous section, finding a feasible solution for the HFFVRP can be quite difficult. Therefore, one way of dealing with this could be to use methods designed on purpose to find feasible solutions. This can be achieved, for example, by giving priority to the customers with higher demand, during the construction of the routes. However, we verified that such methods are not the most appropriate to reach the ultimate purpose that is finding high quality solutions. Consequently, we decided to use a method that is not so efficient in finding feasible solutions, but that yields solutions with a structure that responds better when improved by the TSA. For some test problems, the initial solutions are infeasible, but the TSA has no difficulty in find feasible ones rather quickly.

Initially, a travelling salesman problem (TSP) tour including all the customers is constructed. The tour starts with the depot and each time a new customer is inserted, following their order number, using the GENI algorithm created by Gendreau et al. [17]. The partition of the tour into feasible routes is performed as follows: the partition starts with the customer adjacent to the depot (the tour is represented as a vector whose first element is the depot and, therefore, the customer selected is its second element); then, the free vehicle with the smallest capacity equal to or greater than the customer's demand is selected, and, therefore, the current route is started (each route is assigned to one vehicle); in this route, the customers that are admissible in terms of capacity are included, one by one, following the sequence defined by the tour. When no more customers are admissible, the process is repeated with the shorter tour that results from eliminating the customers already routed. This process ends when there are no more customers in the tour or when there are no more available vehicles. In this last case, each of the unrouted customers is inserted in one of the existing routes where its insertion cost is lowest, without taking into account the capacity of the vehicle.

2.2. Neighbourhood structure

The neighbourhood structure is a key feature in the performance of any tabu search algorithm, because it determines the extent and the quality of the solution space explored. The TSA comprises four types of neighbourhood moves: the first three are insertions (single, double and triple) and the forth is a swap. As we will explain later these moves are not performed in every iteration for two reasons: to diversify the search and to keep the computing time at reasonable levels. In each iteration all the customers are candidates to be moved.

In a single insertion move, a candidate customer x_i is removed from its current route (origin) and a trial insertion is made in any of the other routes (destination), containing at least one of the δ-nearest neighbours of x_i, between any two customers or near the depot. The value of δ varies during the search and will only be defined later on. This δ-neighbourhood restriction not only reduces the computing time but also contributes to find better solutions. Gendreau et al. [21], in their VRP solving algorithm also imposed this neighbourhood condition to allow a customer to be moved to another route. This concept has also been used by several other researchers, but the main difference in relation to them is that in our algorithm there is a variation of the value of δ according to the evolution of the search.

In a double insertion move, the operation is similar to the single insertion one, except for the following: the candidates are any pair of customers (x_i, x_j) belonging to the same route, the destination route should have at least one of the δ-nearest neighbours of x_i or x_j and the insertion place of x_j takes into account also the previous (virtual) insertion of x_i.

In a triple insertion move, a chain or segment of three consecutive customers (x_i, x_j, x_k) belonging to the same route is considered. The destination route should have at least one of the δ-nearest neighbours of x_i or x_j or x_k and the chain is inserted between any two customers or near depot, either in the current order of the chain or in the reverse order, i.e., (x_k, x_j, x_i).

Either the single, the double or the triple insertion can create a new route, subject, however, to the following two conditions: *i)* the origin route should have at least 2, 3 or 4 customers for the single, double and triple insertion, respectively; ii) the type of vehicle of the destination route should be the lowest (free vehicle) capable of containing the demand of the moving customers.

A swap move consists of exchanging two customers belonging to two different routes. A trial swap of a customer $x_i \in r_i$ (route r_i) with a customer $x_j \in r_j$ exists if and only if: *i)* at least one customer of r_j belongs to the δ-neighbourhood of x_i; *ii)* at least one of the routes r_i or r_j contains more than one customer.

In this implementation, the frequencies of the different types of move may change according to the cycle and phase of the algorithm. Parameters F_I, F_D, F_T and F_S denote the frequencies of the single, double and triple insertion and swap moves, respectively. For example, if $F_S = 10$ and $F_T = \infty$, this implies that the swap move is only tested every 10 iterations and the triple insertion is never executed.

The trial move chosen depends on the effect of the move on the objective function defined in the next section.

As we said before, in the designing of the TSA we took advantage of the knowledge that we acquired from our previous tabu search algorithm for the FSMVRP (Brandão [5]). Nevertheless, the experiments have shown that the resolution of the HFFVRP is much more difficult, mainly because it is harder to escape from a local optimum and reach other local optima of higher quality. This was particularly true for a set of problems created by Li et al. [6] because of the symmetry of the data - the customers are all located around the depot forming a set of groups that are placed along the same radius (angle), and half of the customers have a given demand, while the other set of customers have also the same demand, but different from the first set. These difficulties were overcome mainly by the use of several different route improvement procedures, by the use of long term memory and by shaking the solution obtained at the end of each phase.

2.3. Objective function

For any candidate solution, *S'*, the objective function is denoted by *f(S')*. The objective function to be minimized by the TSA is defined by the next equation:

$$f(S') = \sum_{i=1}^{r} (c_i + P l_i). \qquad (1)$$

Where, r is the total number of routes in *S'*, c_i is the variable cost of route i, P is a penalty term and l_i is the load excess in route i.

When long term memory is used, the objective function, $f_1(S')$, is given by the following equation:

$$\begin{cases} f_1(S') = f(S') \Leftarrow f(S') \leq f(S) \\ f_1(S') = f(S') + \beta D\sqrt{rn}\rho/K \Leftarrow f(S') > f(S) \end{cases} \quad (2)$$

Where S is the current solution, β is a scaling parameter, D is the largest absolute difference in $f(S)$ that has occurred between two consecutive iterations, ρ is the number of times that a customer has been moved (if the move originating S' involves more than one customer, it is considered the one with the higher number of moves) and K is the number of iterations executed so far.

Several authors, like Gendreau et al. [21], Gendreau et al. [22] and Hoff et al. [23], have used a diversification strategy similar to this. In our case, β is allowed to vary during the search, instead of using a constant value as these authors do. This diversification strategy has proven to be quite useful with one set of problems, but not with the other set.

The *strategic oscillation* created by allowing the search to cross the boundary between the feasible and the infeasible space plays an important role in the quality of the final solutions. We tried with the same algorithm, but without allowing infeasible solutions to occur, and the results were rather worse.

We should note that for any S the value of l_i depends on the type of vehicle assigned to route i. Therefore, in each trial move, the type of the vehicle can be changed to one with more or less load capacity, and the vehicle type that implies the lowest value of $f(S)$ will be the one chosen. All this takes place in the insertion (single double or triple), as follows. Let us suppose that r_i is the origin route after removing the customer (or customers), and r_j is the destination route after inserting the customer (or customers). If r_i is infeasible nothing is done; otherwise, the vehicle type is reduced while r_i is still feasible. If r_j is feasible nothing is done; otherwise, the vehicle type is increased by one unit at a time until r_j becomes feasible or the type of vehicle with the highest capacity is reached. At the same time, for each vehicle type, including the initial one, the cost of the route is calculated by (1) and, in the end, the vehicle type to which the lowest cost corresponds is chosen. Regarding the trial swap move, after performing the move, the same procedure is applied to both routes r_k and r_m: if r_k (r_m) is feasible or infeasible, it is handled as r_i or r_j in the insertion, respectively. It is important to note that this change of vehicle type only can happen if there is a free vehicle of that type. On the other hand, the number of free vehicles is dynamic in each trial move, i.e., if r_i is assigned to a vehicle of a different type, the original vehicle type of r_i will have one more vehicle free and the destination vehicle type will have one less.

2.4. Admissibility of moves

The tabu status of a move is defined in the following way: a customer that leaves a route cannot return to it during the next θ iterations, where θ is the tabu tenure. Furthermore, when a new route is created, the customers originating it cannot go to any of the existing routes during θ iterations, in order to avoid the destruction of recently formed routes. The tabu tenure changes systematically during the search according to the evolution of the results. The tabu restriction may be overridden if the move produces a solution that is better than the ones found in the past. This is referred to as the aspiration criterion. In the TSA, a trial move to solution *S'* is regarded as admissible if:

i) it is not currently on the tabu list;

ii) it is tabu and infeasible, but the value of *f(S')* is lower than the value of the best infeasible solution yet found;

iii) it is tabu and feasible, but the value of *f(S')* is lower than the value of the best feasible solution yet found.

2.5. Route improvement procedures

In the trial moves, the cost of moving a customer is calculated as a simple insertion. However, the optimal cost of the two modified routes may be much lower, but its determination is prohibitive due to the computing time required, since this will imply to solve a TSP for each route, which is a well known NP-hard problem. Instead of this, at the end of each iteration, we try to improve the two routes just modified by one of the following three post-optimization procedures: *i)* the customers that had been moved are introduced into their new routes by the GENI (note that in this case no real insertion is made during the trial insertion, only the insertion cost is calculated); *ii)* US (another algorithm developed by Gendreau et al. [17]); *iii)* 2-opt. After this, if a new best feasible solution is found in this iteration, the US procedure is applied to each individual route, in order to try to reduce its cost further.

Our experiments indicate that the influence of these procedures in the quality of the final solution depends on the size of the problem and on the geographic distribution of the customers: the GENI performs well for smaller problems with a non uniform distribution (called *set 1*), while alternating the use of these procedures generates better

solutions and faster for large problems (*set 2*), with customers symmetrically located in the space.

2.6. Diversification and intensification

The main sources of diversification in the TSA are: the use of several different phases; the strategic oscillation, as well as the change of the vehicle type driven by the objective function; the long term memory expressed in the modified objective function (2); the shaking of a good solution; the change of the tabu tenure, the neighbourhood and the frequency of the types of moves, during the search. The intensification is mainly achieved using the post-optimization procedures and the restart, after a given number of iterations, with the best found solution. The fact that several parameter values, such as the duration of each cycle, depend on the evolution of the search is also important, as we explain in Section 2.9.

2.7. Shaking

Towards the end of the search, when the best known solution is good, shaking it may be the only way of escaping from a local optimum and reaching better solutions. This technique has been used for a long time in the variable neighbourhood search (VNS) metaheuristic (Hansen and Mladenovic [24]) and it is applied when a local optimum is reached within a given neighbourhood. Shaking a solution *S* consists of generating another solution *S'* in its neighbourhood. In the VNS this is done by performing a random move according to the current neighbourhood structure. Our implementation is rather different because the transformation of *S* is not determined by the neighbourhood structure defined in Section 2.2, i.e., the moves are different from those used in the tabu search, the transformation is deterministic, *S'* is farther from *S* than it would be using these moves, and the shaking is only applied a few times during all the search process.

The same shaking concept, although known as perturbation, has been used by Dueck [16] in his great deluge and record-to-record travel algorithms. However, as far as we know, this the first time that the shaking was applied within a tabu search algorithm.

In the TSA the shaking is applied after a long process of search when the best known solution is already good. This means that, in order to reach other promising regions of the solution space, this solution should suffer a significant transformation; otherwise, in the prosecution of the search, the new solution will be quickly attracted to the original one. On the contrary, according to the proximate optimality principle – POP – (Glover and Laguna [25], pp. 138-141), a too large transformation may put the new solution in a poor region of the solution space from where it is difficult to find good solutions. Therefore, the compromise between these two extremes can be a perturbation large enough to escape from the current local optimum, but keeping most of the current structure unchanged. Among the several types of perturbations tried, including one similar to that used by Li et al. [6], we describe below the one used in the final algorithm, which produced the best results.

The shaking used in the TSA consists of forcing each of the $\lfloor r/2 \rfloor$ routes of S to start with one of the $\lfloor r/2 \rfloor$ nearest customers of the depot, and it is applied in the beginning of the phases defined later. These customers are removed from their present place and inserted one by one in the beginning (just after the depot) of each route, starting with the highest route type and the nearest customer of the depot (the ties are arbitrarily broken). This is done even if the resulting route is infeasible in terms of capacity. If after shaking the best known solution, S, and subsequent tabu search process there is no improvement in S, this implies that the result of the next shaking will be the same. To avoid this occurrence, the next shaking is applied to the r routes of S, if there has been no improvement after the previous shaking. However, we should note, that starting from the same solution does not mean that the search path will be the same, because the phase is different, but by using this different shaking we guarantee a higher diversification.

The rationale behind this shaking procedure is that this way these customers are served at a minimum cost and, therefore, we could expect that good solutions may have this property.

2.8. Detailed description of the TSA

The parameters and counters of iterations not already mentioned, which are required to understand the algorithm are the following:

- Iteration counter – K

- Limit of the total number of iterations in a phase – K_L
- Number of iterations since the beginning of the cycle without improving the best solution (feasible or infeasible) – K_B
- Number of iterations since the best feasible solution was found – K_{BF}
- Limit of K_B in the current cycle – K_{BL}
- Best feasible solution – S_{BF}

Step 1 – Generate the initial solution.

Step 2.1. – Set the initial parameters of the phase – K_L, β, θ, δ.

Step 2.2. – Set the initial parameters of the cycle – P, K_{BL}, F_I, F_D, F_T and F_S and empty the tabu list.

Step 3 – Repeat while the stopping criterion ($K > K_L$ and $K_{BF} > 4K_{BL}$) is not met:

Step 3.1. Perform the trial moves and choose the best admissible.

Step 3.2. Introduce into their respective routes the customers that have been moved using GENI or try to improve the two routes modified using 2-opt or US.

Step 3.3. If the solution found in *Step 3.2* is feasible and better than the best known, try to improve it applying US to all the routes and store the resulting solution as the new best solution. Update tabu list, P, iteration counters and best feasible and infeasible solutions.

Step 3.4. If the middle of the cycle has been reached, i.e., $K_B = K_{BL}/ 2$, change θ, δ, F_I, F_D, F_T and F_S.

Step 3.5. If $K_B = K_{BL}$, or $K_{BF} = 3K_{BL}$ (end of the cycle), change K_{BL}, β, θ, δ, F_I, F_D, F_T and F_S. Set $K_B = 0$, empty the tabu list and restart with S = S_{BF}.

Step 4 – If the established number of phases has been reached, stop; otherwise go to *Step 2.1*.

This algorithm is represented by a flow chart in figure 1. This figure shows that the basic structure of the TSA is the *cycle*, which corresponds to *Step 3* in the previous algorithm.

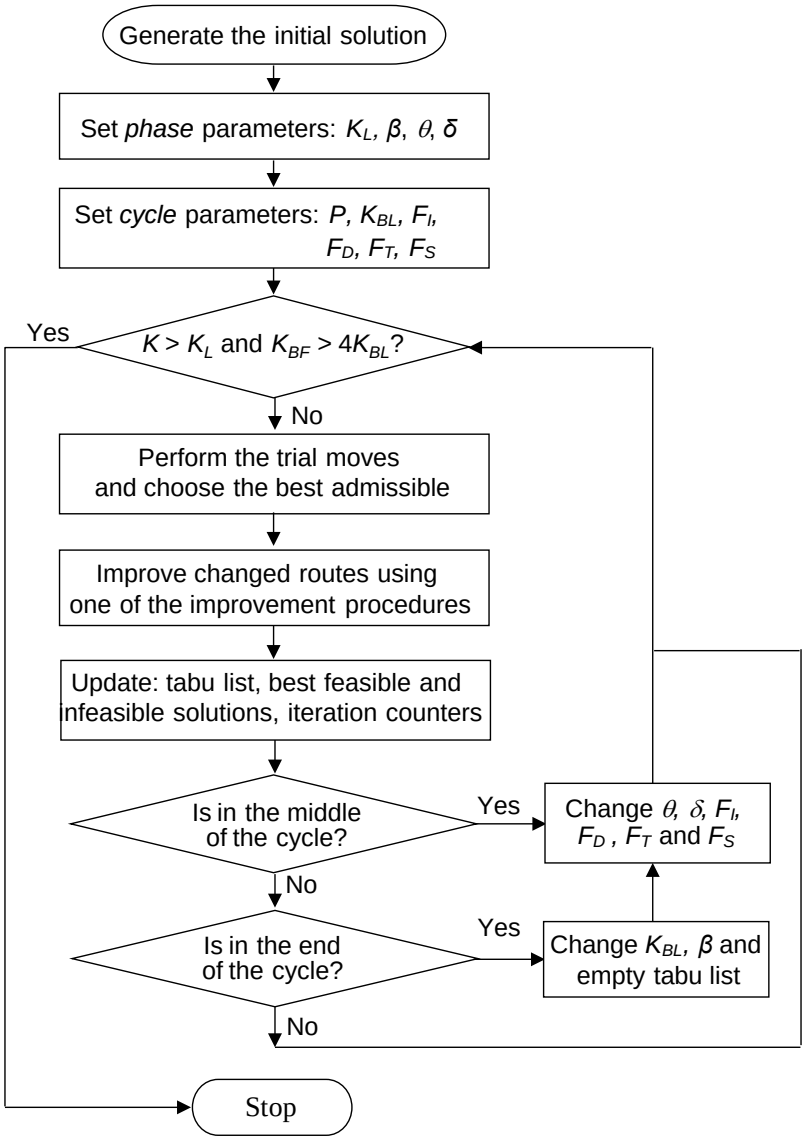

Figure 1 – Flow chart of the TSA.

A cycle ends after a given number of iterations (K_{BL}) without improving the best feasible or infeasible solution (S_B) or when the number of iterations without improving the best feasible solution reaches $3K_{BL}$. Therefore, in spite of K_{BL} being fixed in each cycle, the duration of the cycle varies according to the search, because each time S_B is improved, the counter K_B restarts from zero or the cycle is interrupted if $K_{BF} = 3K_{BL}$. Besides, the value of K_{BL} is increased from one cycle to the next. The middle of the cycle is reached when $K_B = K_{BL}/2$ (note that in each cycle the middle can be reached more than once). A *phase* is a set of cycles, whose number is not known in advance because it depends on the evolution of the search and on the stopping criterion. Any cycle and, therefore, the phase can be interrupted if the stopping criterion is reached. The TSA contains six phases for the set of problems with fewer customers (*set 1*) and three phases for the other set of problems with more customers (*set 2*). In this case, only

three phases were executed in order to keep a good balance between the quality of the solutions and the computing time, in comparison to the algorithms of other authors. *Phase 1* starts with the solution given by the giant tour. The other phases start with the best feasible solution found in the previous phase. However, this solution may suffer a perturbation before the phase begins, as will be explained later on. When solving *set 2*, *phase 1* was executed twice, using two different initial values for the scaling factor (0.013 in one execution and 0.015 in the other), and the best solution found was passed on to the next phase. *Phases 1* to *4*, when applied to *set 1*, only differ in the initial solution, in the initial values of θ and δ and in the limit for the total number of iterations. *Phases 5* and *6*, have as additional difference in relation to the other four, the frequencies of the types of moves: the insertion is less frequent, as we mentioned before, the double insertion is not applied, the swap move is more frequent and the triple insertion is more frequent (in fact it is not used in the previous phases). Due to the higher frequency of the swap moves in these two phases, the average computing time per iteration is higher than in the first four phases. The value of the initial parameters of *phases 5* and *6* is identical to those of phases 1 and 2, respectively. When solving the problems of *set 2*, *phases 1* to *3* present the following differences to when *set 1* is solved: the scaling parameter is different from zero and, therefore, the cost of a trial move is evaluated by (2); in *phase 1*, the procedures 2-opt and US are applied alternately, instead of GENI; in *phase 2* and *3*, these procedures are applied alternately during the first half of the cycle, i.e., while $K_B \leq K_{BL}/2$, and GENI in the second half.

The execution of the trial moves follows the following sequence: single insertion, double insertion, swap and triple insertion, performing all the potential moves of each type and then proceeding to the next. If in a given iteration a trial move generates a solution better than the best already discovered no further trial move is performed in that iteration.

2.9. Parameter tuning

As we explained previously, the TSA has several characteristics in common with an algorithm that we created for the FSMVRP (Brandão [5]). Therefore, in the definition of the value of several TSA parameters we used that previous knowledge. This is the case with θ, δ and the frequencies of the single and double insertion and swap moves, for which similar values were used with the FSMVRP. For setting the

other parameters, we used a set of experiments and principles which will be here described. These experiments were performed with the problems of *set 1*. We should note that there is no way of defining the most effective values of the parameters. Consequently, they were established based on intuition and on those experiments and principles, and the results show that the combinations of values used work well, but, certainly, that better combinations could exist.

First, the most important feature that distinguishes different instances of the HFFVRP is the number of customers, *n*. Therefore, the value of most of the parameters is a function of *n*. This rule ensures to some extent a similar performance of the TSA with problems of different sizes. Another important rule, applied to almost every parameter, is the variation of its value during the search. This aims to induce diversification and its relevance is increased by the fact that the TSA is deterministic.

The two most influential parameters of the TSA are the tabu tenure *(θ)* and the neighbourhood restriction *(δ)*.

The initial value of θ is given by *n/c* (by default, a number is always rounded up to the nearest integer, if the parameter must be integer), where *c* is a constant. The experiments reported in Brandão [5] indicated that c = 2 is the best value. Therefore, we decided to use values around this without further experiments, because of the adaptive nature of θ, which reduces the need of a precise preliminary calibration. During the search θ changes dynamically in the following way: whenever the middle of the cycle is reached, it is set as $\theta = \max(\theta/2, 7)$ and at the end of the cycle it is defined as $\theta = \min(2\theta + 3, 0.6n)$. However, applying only these rules, it may happen that when the change of θ is due, its value remains the same, if it is already in the extremes of those intervals. So, if this happens, it is set $\theta = \theta + 1$, in the middle of the cycle, and $\theta = \theta - 3$, at the end of the cycle. We should note that without any bounds, i.e., if θ is simply set as $\theta = \theta/2$, in the middle of the cycle, and $\theta = 2\theta + 3$, at the end, the average solution cost (for *set 1*) is only very slightly worse. The value of θ is reduced in the middle of the cycle in order to allow intensification, i.e., to explore more thoroughly the current region of the solution space. On the contrary, when the end of the cycle is reached, that means no better solutions could be found in current region and, consequently, more distant regions should be tried. So, θ is increased in order to drive the search towards that regions.

In order to have influence in the search, δ must be small. Its maximum value, δ_L, takes into account the number of customers and the number of vehicles required to serve them. So it was defined as $\delta_L = \min(n\text{-}1, 2c_r)$, where c_r is the number of customers of the route of the initial solution with more customers. As it happens with the tabu tenure, δ also changes dynamically during the search as follows: in the middle of the cycle, $\delta = \min(\delta + 1, \delta_L)$ and at the end of the cycle, $\delta = \min(\delta + 2, \delta_L)$. A set of experiments indicated that the initial values should be very low. In *phase 1* the best average results were obtained using $\delta = 1$, with *set 1*, and $\delta = 2$, with *set 2*. In the latter case, $\delta = 1$ is not a good choice because some routes have a very large number of customers and, therefore, for many customers, all their δ-nearest neighbours are already in the same route. Consequently, during many of the initial iterations only a few moves or no moves at all are allowed, which represents a waste of computing time. The initial value of δ increases slightly in the next phases in order to allow the exploration of different and larger regions of the solution space. The reason for starting with a very low value of δ and increasing it during the search is the following: a low value drives the search towards a promising region of the solution space; once there, when no better solutions could be found, the allowable search space should be enlarged. At the end of the cycle there is a higher diversification in the search space than in the middle of the cycle and, therefore, a higher increment of δ is desirable.

The GENIUS algorithm depends on a parameter, p, that defines the neighbourhood around each customer, and we used $p = 5$ because according to the authors (Gendreau et al. [17]) it gives a good compromise between solution quality and computing time.

The value of P was defined simply according to our previous experience with other types of tabu search algorithms. Initially $P = 1$. If for 10 consecutive iterations, all the solutions are infeasible, P is multiplied by 2. Conversely, if all the solutions are feasible for the last 10 consecutive iterations, P is divided by 2.

The restart associated with the cycle plays an important role in the performance of the TSA because it promotes intensification around the best solution found and some diversification is also guaranteed by changing parameters' values. If the cycles are too long there is not enough time for restarting and if they are too short the search is limited to a narrow region of the solution space. According to our experiments, good results are obtained when the cycle limit (K_{BL}) is around $15n$. This limit is increased by $2n$ at every restart to allow some more time to find better solutions because this becomes harder as

the search progresses. With the problems of *set 2*, better results could be found using a higher value for this limit, so $K_{BL} = 30n$ was used instead. As stated in Section 2.8 (step 3.5), the restart is done with the solution S_{BF}. The first move after the restart cannot be the same as the move performed immediately after discovering S_{BF}. This rule reinforces the guarantee, already given by the change of parameters' values, that the search path followed after the restart is different from the path followed after S_{BF} was found.

The execution of each phase of the TSA is stopped when the limits of the total number of iterations, K_L, and number of iterations without improving the best feasible solution, K_{BFL}, are both reached. In the definition of these limits we tried to obtain a good compromise between solution quality and computing time. Furthermore, K_{BFL} is the quadruple of K_{BL} in order to allow the search to continue during a reasonable number of iterations after the restart. The value of K_L is higher for *phase 1* than for the other phases because, usually, these phases contribute much less to improve the solution. Therefore, in general, would be a waste of computing time if the value of K_L was identical to that of *phase 1*. The other component of the stopping criterion (K_{BFL}) is adapted to the evolution of the search, allowing this to go on whenever new best feasible solutions are discovered, because the corresponding iteration counter, K_{BF}, restarts from zero when this happens. Consequently, K_{BFL} also compensates, to some extent, a less appropriate choice of K_L.

In order to understand the frequencies defined for the different types of moves, we should, firstly, observe their properties. On average, among the four types of moves, the swap is one that causes the lowest absolute variation in the use of the capacity of the two vehicles involved in each iteration. Among the four types of moves used, the trial single insertion is the one that causes the lowest transformation in a solution and that allows a more detailed exploration of the solution space. In relation to these two aspects the other moves are ordered as follows: swap, double insertion and triple insertion. Besides, any transformation produced by the other three types of moves can be reached by the trial single insertion, although requiring several iterations. This is why the trial single insertion moves are performed in every iteration, except in *phases 5* and *6*, where they are applied only if none of the others is executed (in these two phases more dramatic changes are required). For the same reasons, the swap move is the next with higher frequency. As a rule, these two types of trial moves (double insertion and swap) are not performed in the same iteration so that they do not compete with each other. The

variation of the frequencies during the cycle induces a diversification in the search. Besides, it should also be noted that the swap moves and the triple insertion are more powerful in breaking the structure of a given solution, this is why they are so frequent and so important in *phases 5* and *6*, when the initial solution is already good. We observed that both the double insertion and the swap moves give an important contribution for discovering good solutions. However, if their frequency is very high (for example, every iteration) not only the computing time increases considerably but also the final results are worse. This behaviour may be explained as follows: these moves drive the search towards a region of the solution space which is rather distant from the current one and, if this happens frequently, a high perturbation is created acting against the POP. However, as we explained before, in *phases 5* and *6*, this higher perturbation is really required. Conversely, the single insertion move allows the exploration of the same "level", according to the concept defined by the POP. We should note that the perturbation created by those types of moves is boosted by the existence of different types of vehicles.

The long term memory, as expressed by (2), played an important role in finding good solutions for the problems of *set 2*. However, the results are slightly worse if it is used with *set 1* and, therefore, it was only applied to *set 2*. This can easily be explained by the fact that too much diversification is not beneficial for the results and, due to the different characteristics of the data of both sets, *set 2* requires much more diversification than *set 1*. We tried values of $\beta \in [0.01, 0.015]$ and the average results obtained were similar, being a little better the average solutions found with $\beta = 0.015$. Nevertheless, the individual solutions vary significantly with the value used. Due to this we established the following: *i)* Start the phase with a given initial value β and then increase it by 0.001 at each restart of the cycle, but never allowing it to be greater than 0.015; when this happens its value is set at 0.012 and increased again by the same amount at the end of the cycle. So, the value of β oscillates between 0.012 and 0.015; *ii)* Repeat *phase 1* with the initial values of $\beta = 0.013$ and $\beta = 0.015$, as explained before. In each cycle the succession of solutions is related to the solution starting the cycle and, by definition, D results from these solutions. Consequently, we set $D = 0$ at the beginning of each cycle.

The shaking, as defined in the TSA, produces a large perturbation (in general, the resulting solution is infeasible) and, consequently, the following two conditions have been established for applying it: *i)* the solution should be already good and,

therefore, very difficult to improve by other means; *ii)* the algorithm should have enough iterations to recover from this perturbation and find a path of good solutions. Due to this, the shaking was only applied at the beginning of *phases 4* and *6*, with *set 1*, and at the beginning of *phases 2* and *3* with *set 2*.

The value used for the main parameters can now be summarized as follows:

● The initial value of θ is the following: $n/2$ in phases 1, 3 and 5; $n/3$ in phases 2 and 6 and $n/2.5$ in phase 4.

● The initial value of δ is the following: 1 (2 with set 2) in phases 1 and 5; 2 in phases 2 and 6; 5 in phase 3 and 10 in phase 4.

● $K_{BL} = 15n$ if $n < 200$ and $K_{BL} = 30n$ if $n \geq 200$. K_{BL} is increased by $2n$ at the end of the cycle.

● $K_L = 3000\sqrt{n}$ in *phases 1* and *6*, and in the other phases $K_L = 2000\sqrt{n}$.

● In the first four phases: $F_D = 10$ and $F_S = 5$, at the beginning of the cycle; $F_D = 5$ and $F_S = 10$, in the middle of the cycle; $F_T = \infty$. In *phases 5* and *6*: $F_S = 1$, at the beginning of the cycle; $F_S = 2$, in the middle of the cycle; $F_D = \infty$ and $F_T = 7$.

● β oscillates between 0.012 and 0.015, starting with $\beta = 0.012$ and increasing 0.001 at the end of each cycle. This is different in *phase 1* because it is repeated and, in this case, the initial value of β is the following: 0.013 in the first execution and 0.015 in the second one.

3. Computational experiments

3.1. Benchmark problems from the literature

The TSA has been programmed in C language and executed on a Compaq Presario X1000 Intel Pentium M at 1.4 GHz, 512 MB of RAM. The performance of our algorithm was tested using two sets of benchmark problems from the literature, whose specific data for the HFFVRP are in the Tables 1 and 2. The first set (*set 1*, Table 1) was taken from Taillard [1] and contains eight problems, numbered from 13 to 20. The second set (*set 2*) contains five problems created by Li et al. [6], identified as H1 to H5. There are two important differences in the data between these two sets of problems: *i)* the number of customers in *set 1* is between 50 and 100 customers, while in *set 2* it goes from 200 to 360 customers; *ii)* the demand and the coordinates of *set 1* were originally taken from Christofides and Eilon [26] and have been randomly generated, while in *set*

2 there are only two different values for the demand and all the customers are located at symmetric positions around the depot in concentric circles, which means that their distribution is much more uniform. These two differences have important implications in the resolution.

In *set 2* (p. 2738, Table 5 of Li et al.[6]) we found some typographical errors: *i)* in problem H2, one vehicle of type E is missing, otherwise the available capacity is not enough for the demand; in problem H3, % (100 × (total demand/total capacity)) = 93.33 and not 94.76; *iii)* In problem H5, we suspect, but we could not confirm it with the authors, that the variable cost per unit of distance travelled for each type of vehicle, is not the value written there, but the same as the corresponding one for H4. The reason for this hypothesis is that we found better solutions than Li et al. [6] for H1 to H4, and a substantially worse solution for H5. However, assuming identical unit variable costs to those of H4, our best solution is a little better. In spite of this remark, we put in Table 2 the same unit variable costs that appear in Table 5 of Li et al. [6] and we solved the problem with these costs.

[Insert Tables 1 and 2]

We compare the results produced by our algorithm with those given by the algorithms of Taillard [1], Tarantilis et al. [12] (the other algorithm by these authors [11], is not used because the results are slightly worse) and Li et al. [6]. The first algorithm was executed on Sun Sparc workstation at 50 MHz, while the second was executed on a Pentium II at 400 MHz, 128 MB of RAM, and Li et al. [6] used an Athlon at 1 GHz, 256 MB of RAM. Although being aware of the difficulties and limitations in comparing the speeds of the different computers, we use Dongarra [27] tables to assume that the speed of these computers, measured in millions of floating-point operations per second (Mflop/s), is the following: workstation - 10 Mflop/s, Pentium II – 80 Mflop/s, Athlon – 450 Mflop/s and Pentium M – 250 Mflop/s.

The authors used different ways of presenting their results and solved different problems: Taillard [1] executed the algorithm five times, giving the average solutions and computing times; Tarantilis et al.'s [12] algorithm contains several random parameters, but they only provide the results for one run; Li et al.'s [6] algorithm is

deterministic as ours, and they have solved both sets of problems, while the other authors only solved *set 1*.

The tables give the following information: the number of customers; the *best known* solution costs from the literature, with the indication of the source in the case the solution was found by only one algorithm; "*our best*" which is the best result obtained by our algorithm during all experiments with different values of the parameters; the *cost* of the solution produced by the different algorithms and respective computing time in seconds (*CPU*) on the computers previously indicated; the *average* for each set of problems; the percentage difference of the average in relation to the average of the best known solution costs (*Av. deviation*)*;* the number of solutions found by each algorithm which are identical or better than the best published ones (*NB*).

[Insert Tables 3 and 4]

The results in Table 3 show that the TSA was able to find all the best known solutions except two. In comparison with Taillard's [1] algorithm, the TSA produces better solutions for all the problems, but his algorithm is about twice as fast. In relation to Tarantilis et al. [12], we can see that the solutions given by the TSA are a little better and the computing time is a bit lower. On average, Li et al.'s [6] algorithm produced slightly better solutions than the TSA and one more solution identical to the best known, but their computing time is about three times higher.

It is worth mentioning that at the end of *phase 4* the average solution cost is 0.51% above the best known. This means that the improvement in the last two *phases* (*5* and *6*) has been only 0.4%, but the average computing time taken by these phases was about 80 seconds, i.e., more than half of the average total computation time. This behaviour, which is typical of this kind of algorithms, means that a high price, in terms of computing time, has to be paid for a small improvement in the solution when it is already good.

The solution cost for the problem H5 in Table 4 contains two values: one assuming that the unit variable costs in Table 2 are correct and another, between brackets, assuming that these costs are identical to those of the problem H4. We should note that this second cost could be better if the problem was solved assuming the unit variable costs of H4. The results in Table 4, excluding the problem H5 because of the doubts just mentioned, show that the TSA gives better solutions than Li et al.'s [6]

algorithm for every problem and found new best solutions for all the problems. However, the computing time taken by the TSA is almost the double.

Overall, the TSA found six solutions equal to the best known and four new best solutions, while the algorithm of Li et al.[6] found only seven identical to the best known. By using different combinations of parameters during our experiments, the TSA was able to find all the best known solutions but one, and four new best ones for the 12 problems.

3.2. New test problems

In order to increase the statistical significance of the computational experiments we created five new test problems designated as N1 to N5, set 3. The data for the depot and the customers (demand and coordinates) were taken from VRP examples, as follows: from Christofides et al. [28], problems 4-5 and 11-12, for N1 to N4, respectively; from Fisher [29], problem 12, for N5. In the first two problems (N1 and N2), the coordinates were randomly generated in the plane, while in problems N3 and N4, the customers appear in clusters. Problem N5 is a real problem. The characteristics of the vehicle fleet are in Table 5. These characteristics have been established in a way similar to that used by Taillard [1], p. 8, namely: i) the variable cost, v_k, have been chosen in such a way that good solutions for the FSMVRP are composed of vehicles of each type; the number of vehicles of each type, n_k, have been defined in such a way that the fleet composition of good FSMVRP solutions is very different from that of good HFFVRP solutions; iii) the total capacity is such that any HFFVRP solution must have at least one vehicle of each type (not all Taillard [1] or Li et al. [6] problems satisfy this condition). The average capacity ratio of our set of problems is slightly higher than that of the other two sets.

[Insert Table 5]

The quality of the TSA solutions for sets 1 and 2 was evaluated by comparing with the solutions produced by other algorithms. However, another way of measuring it is comparing the solution cost with the cost of the lower bound. The exact FSMVRP solution is a lower bound for the HFFVRP solution. We do not have the exact FSMVRP solutions for the three sets of problems under consideration, but we can calculate very good FSMVRP solutions, using the algorithm of Brandão [5]. So, using these “lower

bounds", we can say that, very likely, our best solutions for the sets 1 and 2 (Tables 3 and 4) are, on average, no more than 1.3% and 12.3% above optimum, respectively. Certainly, this gap is shared by the "lower bound" and the upper bound.

For the new problems the results are in Table 6, where the solution cost and the respective vehicle fleet are presented.

[Insert Table 6]

These results show that the average gap between the "lower bound" and the upper bound is 2.54%. As expected, due to the relative dimension of the three sets of problems, the gap for this set is in the middle of the other two sets. However, we are well aware that there are other relevant factors influencing this gap, like the cost structure and the ratio total demand versus total capacity of the vehicles. The solutions obtained for this new set of problems confirm the good performance of the TSA.

4. Conclusions

So far as we know, the TSA was the first tabu search algorithm applied to the heterogeneous fixed fleet vehicle routing problem. The results have proven that this metaheuristic is appropriate and the algorithm is able to find high quality solutions in a reasonable computing time. Four new best solutions are provided for the test problems that have been studied by other researchers.

The solutions presented demonstrate the good performance of the TSA when compared to the algorithms of Taillard [1], Tarantilis et al. [12] and Li et al. [6]. The algorithm of Li et al. [6] is also deterministic, but the performance of the TSA is slightly better. In general, all these algorithms yield good results, but Taillard's [1] algorithm, in spite of its good performance, is not appropriate for problems of large dimensions, as the author recognises, because it requires the exact resolution of a set partitioning problem, which is an NP-hard problem.

The main conclusions that we could draw from the computational experiments are the following: the problems of large dimension impose several additional difficulties, especially, if the data structure is uniform; the δ-neighbourhood plays an important role both in the quality of the solutions and in the computing time; the use of long term memory in the algorithm can diversify significantly the search and enhance the solution quality; a significant shaking can also have a rather positive influence in the

final results; the resolution of the HFFVRP is substantially more difficult than the FSMVRP. We believe that the TSA can easily be adapted to deal efficiently with the HFFVRP containing other additional constraints usually found in real-life. This conviction is reinforced by the fact that the TSA contains several features in common with the algorithm applied to a different type of problem, the FSMVRP (Brandão [5]), where it has proved to have a good performance.

Table 1
Data for the problems of *set 1*

Problem	n	Type of vehicle																	Ratio (%)
		A			B			C			D			E			F			
		Q_A	v_A	n_A	Q_B	v_B	n_B	Q_C	v_C	n_C	Q_D	v_D	n_D	Q_E	v_E	n_E	Q_F	v_F	n_F	
13	50	20	1.0	4	30	1.1	2	40	1.2	4	70	1.7	4	120	2.5	2	200	3.2	1	95.39
14	50	120	1.0	4	160	1.1	2	300	1.4	1										88.45
15	50	50	1.0	4	100	1.6	3	160	2.0	2										94.76
16	50	40	1.0	2	80	1.6	4	140	2.1	3										94.76
17	75	50	1.0	4	120	1.2	4	200	1.5	2	350	1.8	1							95.38
18	75	20	1.0	4	50	1.3	4	100	1.9	2	150	2.4	2	250	2.9	1	400	3.2	1	95.38
19	100	100	1.0	4	200	1.4	3	300	1.7	3										76.74
20	100	60	1.0	6	140	1.7	4	200	2.0	3										95.92

Ratio = 100 × (total demand/total capacity).

Table 2
Data for the problems of *set 2*

Problem	n	Type of vehicle																	Ratio (%)
		A			B			C			D			E			F			
		Q_A	v_A	n_A	Q_B	v_B	n_B	Q_C	v_C	n_C	Q_D	v_D	n_D	Q_E	v_E	n_E	Q_F	v_F	n_F	
H1	200	50	1.0	8	100	1.1	6	200	1.2	4	500	1.7	3	1000	2.5	1				93.02
H2	240	50	1.0	10	100	1.1	5	200	1.2	5	500	1.7	4	1000	2.5	1				96.00
H3	280	50	1.0	10	100	1.1	5	200	1.2	5	500	1.7	4	1000	2.5	2				93.33
H4	320	50	1.0	10	100	1.1	8	200	1.2	5	500	1.7	2	1000	2.5	2	1500	3	1	94.12
H5	360	50	1.0	10	100	1.2	8	200	1.5	5	500	1.8	1	1500	2.5	2	2000	3	1	92.31

Ratio = 100 × (total demand/total capacity).

Table 3
Comparison of the results for set 1

Problem	*n*	Best Known	Our Best	Taillard		Tarantilis et al.		Li et al.		TSA	
		Cost	Cost	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU
13	50	1517.84[a]	**1517.84**	1536.55	473	1519.96	843	**1517.84**	358	**1517.84**	56
14	50	607.53[a]	**607.53**	623.05	575	611.39	387	**607.53**	141	**607.53**	55
15	50	1015.29	**1015.29**	1022.05	335	**1015.29**	368	**1015.29**	166	**1015.29**	59
16	50	1144.94[a]	**1144.94**	1159.14	350	1145.52	341	**1144.94**	188	**1144.94**	94
17	75	1061.96[a]	**1061.96**	1095.01	2245	1071.01	363	**1061.96**	216	**1061.96**	206
18	75	1823.58[a]	**1823.58**	1894.73	2876	1846.35	971	**1823.58**	366	1831.36	198
19	100	1117.51[b]	1120.33	1156.93	5833	1123.83	428	1120.34	404	1120.34	243
20	100	1534.17[a]	**1534.17**	1592.16	3402	1556.35	1156	**1534.17**	447	**1534.17**	302
Average		1227.85	1228.21	1259.95	2011	1236.21	607	1228.21	286	1229.18	152
Av. deviation (%)			0.03	2.61	–	0.68	–	0.03	–	0.11	–
NB			7	0	–	1	–	7	–	6	–

[a] Solution cost taken from Li et al. [6]
[b] Solution cost taken from Taillard [1].
Values in boldface – identical to the best known.

Table 4
Comparison of the results for *set 2*

Problem	*n*	Best Known	Our Best	Li et al.		TSA	
		Cost	Cost	Cost	CPU	Cost	CPU
H1	200	12067.65	***12050.08***	12067.65	688	***12050.08***	1395
H2	240	10234.40	***10208.32***	10234.40	995	***10226.17***	3650
H3	280	16231.80	***16223.39***	16231.80	1438	***16230.21***	2822
H4	320	17576.10	***17458.65***	17576.10	2256	***17458.65***	8734
H5	360	21850.40	23166.56 (21757.26)	21850.40	3277	23220.72 (21852.36)	13321
Average[a]		14027.49	13985.11	14027.49	1344	13991.28	4150
Av. deviation (%)			-0.30	0.00	–	-0.26	–
NB			4	0	–	4	–

[a] This average is only for the first four problems.
Values in boldface – identical to the best known or better (in this last case they are also in italic).

Table 5
Data for the problems of *set 3*

Problem	n	Type of vehicle																		Ratio (%)
		A			B			C			D			E			F			
		Q_A	v_A	n_A	Q_B	v_B	n_B	Q_C	v_C	n_C	Q_D	v_D	n_D	Q_E	v_E	n_E	Q_F	v_F	n_F	
N1	150	50	1	5	100	1.5	4	150	1.9	4	200	2.2	3	250	2.6	2				95.11
N2	199	50	1	8	100	1.5	6	150	1.9	5	200	2.2	4	250	2.6	2	350	3.2	1	93.71
N3	120	50	1	6	100	1.5	3	150	1.9	3	200	2.2	2							94.83
N4	100	50	1	4	120	1.6	4	180	2.1	4	240	2.6	2							96.28
N5	134	900	1	5	1500	1.5	3	2000	1.8	2	2500	2.2	1							94.32

Ratio = 100 × (total demand/total capacity).

Table 6
The results for *set 3*

Problem	*n*	FSMVRP solution		HFFVRP solution		Gap (%)
		Cost	Vehicle fleet	Cost	Vehicle fleet	
N1	150	2220.01	5A, 2B, 3C, 7D	2243.76	4A, 4B, 4C, 3D, 2E	1.07
N2	199	2827.76	2A, B, 4C, 12D	2874.13	5A, 6B, 5C, 4D, 2E, F	1.64
N3	120	2234.57	2A, 3B, 5D	2386.90	5A, 3B, 3C, 2D	6.82
N4	100	1822.78	3B, 3C, 4D	1839.22	3A, 4B, 4C, 2D	0.90
N5	134	2016.79	7A, B, C, 2D	2062.48	5A, 3B, 2C, D	2.27
Average		–	–	–	–	2.54

Acknowledgements

This research was partially supported by Fundação para a Ciência e Tecnologia under the project nº PTDC/EGE-GES/104092/2008. This support is gratefully acknowledged. We also thank Filomena Louro from the University of Minho Editing Program for revising the text. We also thank two unknown referees for their valuable comments.

Appendix – New best solutions

Here are presented the best solutions found by the TSA for the problem sets 1 and 2. These solutions as well as the solutions for the set 3 and the data can also be found at http://cemapre.iseg.utl.pt/~sbrandao/HFFVRP_solutions_and_data.pdf. All the calculations have been performed with a precision of 64 bits and the total solution cost is presented with four decimal places. In order to save some space, a line can contain more than one route and the following format is used: <route number>: <route> <load> <vehicle type>; .

Problem 13: Solution cost = 1517.8366

1: 0-17-0 20 A; 2: 0-26-0 18 A; 3: 0-16-0 19 A; 4: 0-6-0 19 A; 5: 0-2-0 26 B; 6: 0-4
-0 30 B; 7: 0-40-0 33 C; 8: 0-31-25-0 39 C; 9: 0-7-35-19-0 40 C; 10: 0-27-13-15-0 37
C; 11: 0-34-46-8-0 62 D; 12: 0-49-24-18-50-0 67 D; 13: 0-1-43-42-41-23-0 68 D; 14: 0
-28-22-33-0 68 D; 15: 0-14-11-38-10-0 118 E; 16: 0-12-39-9-32-44-3-0 117 E; 17: 0-30
-48-21-47-36-20-37-5-29-45-0 192 F

Problem 14: Solution cost = 607.5290

1: 0-12-25-50-18-24-44-3-0 120 A; 2: 0-4-34-46-35-7-26-0 119 A; 3: 0-33-1-43-42-41-23
-49-16-0 119 A; 4: 0-30-48-21-28-22-2-6-0 156 B; 5: 0-27-13-15-20-37-36-47-5-29-45-0
159 B; 6: 0-17-40-32-9-39-31-10-38-11-14-19-8-0 300 C

Problem 15: Solution cost = 1015.2939

1: 0-5-38-46-0 41 A; 2: 0-26-7-43-24-0 47 A; 3: 0-37-42-40-19-4-0 47 A; 4: 0-48-23-6
-0 48 A; 5: 0-49-10-39-33-45-15-44-17-0 99 B; 6: 0-12-47-18-0 95 B; 7: 0-41-13-25-14
-0 99 B; 8: 0-11-16-50-9-30-34-21-29-2-32-0 156 C; 9: 0-27-8-31-28-3-36-35-20-22-1-0
145 C

Problem 16: Solution cost = 1144.9360

1: 0-12-0 29 A; 2: 0-7-43-24-0 40 A; 3: 0-37-17-42-19-40-41-4-0 77 B; 4: 0-27-48-23-

6-0 63 B; 5: 0-11-16-50-9-38-46-0 75 B; 6: 0-44-15-45-33-39-10-0 78 B; 7: 0-14-25-13
-18-47-0 138 C; 8: 0-1-22-20-35-36-3-28-31-26-8-0 137 C; 9: 0-32-2-29-21-34-30-49-5-
0 140 C

Problem 17: Solution cost = 1061.9570

1: 0-4-75-0 50 A; 2: 0-34-46-0 46 A; 3: 0-67-26-0 48 A; 4: 0-40-44-24-49-16-6-0 120
B; 5: 0-51-3-50-18-55-25-31-12-0 120 B; 6: 0-74-21-61-28-2-68-0 118 B; 7: 0-8-19-54-
13-57-15-27-52-0 117 B; 8: 0-73-1-62-22-64-42-43-41-56-23-63-33-0 196 C; 9: 0-45-29-
5-37-20-70-60-71-69-36-47-48-30-0 199 C; 10: 0-7-35-53-14-59-11-66-65-38-10-58-72-39
-9-32-17-0 350 D

Problem 18: Solution cost = 1823.5801

1: 0-6-0 19 A; 2: 0-75-0 20 A; 3: 0-25-55-31-0 46 B; 4: 0-34-67-0 49 B; 5: 0-73-56-23
-63-0 49 B; 6: 0-52-46-0 46 B; 7: 0-27-15-57-13-54-19-8-0 98 C; 8: 0-30-48-47-21-74-
0 99 C; 9: 0-62-22-64-42-41-43-1-33-0 147 D; 10: 0-17-3-44-50-18-24-49-16-51-0 146 D;
11: 0-4-45-29-5-37-20-70-60-71-36-69-61-28-2-68-0 248 E; 12: 0-26-7-35-53-14-59-11-66
-65-38-10-58-72-39-9-32-40-12-0 397 F

Problem 19: Solution cost = 1120.3438

1: 0-12-80-68-29-24-25-55-54-0 99 A; 2: 0-60-83-8-46-45-17-84-5-99-96-6-0 98 A; 3: 0
-87-42-14-38-43-15-57-2-0 96 A; 4: 0-76-77-3-79-78-34-35-65-71-9-51-81-33-50-0 199 B;
5: 0-31-88-62-10-63-90-32-66-20-30-70-1-69-27-0 199 B; 6: 0-52-7-19-11-64-49-36-47-48
-82-18-89-0 193 B; 7: 0-53-58-40-21-73-72-74-22-41-75-56-23-67-39-4-26-28-0 278 C; 8:
0-94-95-59-93-85-61-16-86-44-91-100-98-37-92-97-13-0 296 C

Problem 20: Solution cost = 1534.1666

1: 0-11-64-49-36-46-0 57 A; 2: 0-18-83-8-45-17-84-60-0 60 A; 3: 0-54-24-29-34-78-50-
0 60 A; 4: 0-97-92-91-38-43-15-57-2-0 60 A; 5: 0-53-0 14 A; 6: 0-73-74-22-41-58-0 58
A; 7: 0-33-81-9-35-71-65-66-20-51-0 140 B; 8: 0-88-62-19-47-48-82-7-52-0 138 B; 9: 0
-28-76-77-3-79-68-80-12-0 140 B; 10: 0-31-10-63-90-32-30-70-1-69-27-0 137 B; 11: 0-94
-95-59-98-37-100-85-93-99-96-6-0 196 C; 12: 0-89-5-61-16-86-44-14-42-87-13-0 200 C;
13: 0-26-4-55-25-39-67-23-56-75-72-21-40-0 198 C

Problem H1: Solution cost = 12050.0761

1: 0-2-1-20-0 50 A; 2: 0-5-4-3-0 50 A; 3: 0-11-10-9-7-0 100 B; 4: 0-19-18-17-15-0 100
B; 5: 0-105-125-145-165-185-184-164-144-124-104-0 100 B; 6: 0-23-24-44-64-84-85-65-25
-0 100 B; 7: 0-8-27-26-6-0 100 B; 8: 0-14-34-35-16-0 100 B; 9: 0-30-49-69-89-109-129
-149-169-189-188-168-148-128-108-88-68-48-28-0 200 C; 10: 0-32-53-73-93-113-133-153-
173-193-192-172-152-132-112-92-72-52-31-0 200 C; 11: 0-38-57-77-97-117-137-157-177-197
-196-176-156-136-116-96-76-56-36-0 200 C; 12: 0-39-60-80-100-120-140-160-180-200-181
-161-141-121-101-81-61-41-21-0 200 C; 13: 0-13-33-54-74-94-114-134-154-174-194-195-175
-155-135-115-95-75-55-0 500 D; 14: 0-29-50-70-90-110-130-150-170-190-191-171-151-131

-111-91-71-51-12-0 500 D; 15: 0-40-59-79-99-119-139-159-179-199-198-178-158-138-118-98-78-58-37-0 500 D; 16: 0-47-67-87-107-127-147-167-187-186-166-146-126-106-86-66-46-45-43-63-83-103-123-143-163-183-182-162-142-122-102-82-62-42-22-0 1000 E

Problem H2: Solution cost = 10208.3088

1: 0-25-24-23-0 50 A; 2: 0-16-14-13-0 50 A; 3: 0-12-11-9-0 50 A; 4: 0-29-28-27-0 50 A; 5: 0-20-21-22-0 50 A; 6: 0-33-34-36-0 50 A; 7: 0-40-1-2-3-4-5-0 100 B; 8: 0-41-80-79-78-77-76-0 100 B; 9: 0-19-18-17-15-0 100 B; 10: 0-39-38-37-35-0 100 B; 11: 0-10-8-7-6-0 100 B; 12: 0-96-97-137-136-135-134-133-132-92-93-94-95-0 200 C; 13: 0-112-152-192-193-233-232-231-230-229-228-188-189-149-148-108-68-0 200 C; 14: 0-111-151-191-190-150-110-109-69-0 200 C; 15: 0-101-141-140-180-181-182-185-186-146-145-105-106-0 200 C; 16: 0-104-144-184-224-225-226-227-187-147-107-0 200 C; 17: 0-88-128-168-208-209-210-211-212-213-214-215-216-217-218-219-220-221-222-223-183-143-142-102-103-0 500 D; 18: 0-85-125-126-166-165-164-163-162-161-200-199-198-197-196-195-155-156-157-158-159-160-121-122-123-124-84-0 500 D; 19: 0-100-99-98-138-139-179-178-177-176-175-174-173-172-171-170-169-129-130-131-91-90-89-0 500 D; 20: 0-113-153-154-194-234-235-236-237-238-239-240-201-202-203-204-205-206-207-167-127-87-86-0 500 D; 21: 0-32-31-30-70-71-72-73-74-75-114-115-116-117-118-119-120-81-82-83-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62-63-64-65-66-67-26-0 1000 E

Problem H3: Solution cost = 16223.3905

1: 0-23-24-25-0 50 A; 2: 0-4-3-1-0 50 A; 3: 0-6-37-38-39-0 100 B; 4: 0-168-196-224-252-280-253-225-197-169-141-0 100 B; 5: 0-165-193-221-249-277-276-248-220-192-164-0 100 B; 6: 0-84-112-85-86-58-57-0 100 B; 7: 0-80-108-136-137-109-81-82-29-0 100 B; 8: 0-44-72-100-128-156-184-212-240-268-269-241-213-185-157-129-101-73-46-0 200 C; 9: 0-96-124-152-180-208-236-264-265-237-209-181-153-125-97-70-42-0 200 C; 10: 0-47-76-104-132-160-188-216-244-272-273-245-217-189-161-133-105-77-49-0 200 C; 11: 0-32-60-88-116-144-172-200-228-256-257-229-201-173-145-117-89-61-34-0 200 C; 12: 0-36-65-93-121-149-177-205-233-261-260-232-204-176-148-120-92-64-35-0 200 C; 13: 0-40-68-67-95-123-151-179-207-235-263-262-234-206-178-150-122-94-66-0 500 D; 14: 0-43-71-99-127-155-183-211-239-267-266-238-210-182-154-126-98-69-41-0 500 D; 15: 0-48-75-103-131-159-187-215-243-271-270-242-214-186-158-130-102-74-45-0 500 D; 16: 0-63-91-119-147-175-203-231-259-258-230-202-174-146-118-90-62-33-5-0 500 D; 17: 0-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-50-78-106-134-162-190-218-246-274-275-247-219-191-163-135-107-79-51-52-53-54-55-27-26-0 1000 E; 18: 0-28-56-83-111-110-138-166-194-222-250-278-279-251-223-195-167-139-140-113-114-142-170-198-226-254-255-227-199-171-143-115-87-59-31-30-2-0 1000 E

Problem H4: Solution cost = 17458.6474

1: 0-24-25-26-0 50 A; 2: 0-41-80-79-0 50 A; 3: 0-81-121-201-241-281-320-280-240-160-120-0 100 B; 4: 0-137-177-217-257-256-216-176-136-96-56-0 100 B; 5: 0-8-9-10-11-12-13-0 100 B; 6: 0-133-173-213-253-293-292-252-212-172-132-0 100 B; 7: 0-88-128-208-248-288-289-249-209-129-89-0 100 B; 8: 0-125-165-205-245-285-284-244-204-164-124-0 100 B; 9: 0-44-83-84-85-86-45-0 100 B; 10: 0-74-75-77-78-0 100 B; 11: 0-76-157-197-237-277-317-316-315-275-276-236-196-156-155-0 200 C; 12: 0-65-105-104-144-145-225-265-305-306-307-308-268-228-188-148-108-0 200 C; 13: 0-140-180-220-260-300-299-259-219-179-139-0 200 C; 14: 0-73-153-152-192-193-233-272-312-311-310-309-269-229-189-149-68-0 200 C;

15: 0-224-264-304-303-263-262-302-301-261-221-181-141-0 200 C; 16: 0-69-109-150-151-191-190-230-270-271-231-232-273-313-314-274-234-235-195-194-154-0 500 D; 17: 0-57-97-138-178-218-258-298-297-296-295-294-254-255-215-214-174-175-135-134-53-0 500 D; 18: 0-54-55-95-94-93-92-91-90-130-131-171-211-251-291-290-250-210-170-169-168-167-207-247-287-286-246-206-166-126-127-87-46-47-48-49-50-51-52-0 1000 E; 19: 0-43-42-82-122-123-163-203-243-283-282-242-202-162-161-200-199-239-279-319-318-278-238-198-158-159-119-118-117-116-115-114-113-112-111-110-70-71-72-0 1000 E; 20: 0-14-15-16-17-18-19-20-21-22-23-64-63-62-61-60-59-58-98-99-100-101-102-103-143-142-182-222-223-183-184-185-186-226-266-267-227-187-147-146-106-107-67-66-27-28-29-30-31-32-33-34-35-36-37-38-39-40-1-2-3-4-5-6-7-0 1500 F

Problem H5: Solution cost = 23166.5628 (21757.2576)

1: 0-208-244-280-316-352-353-317-281-245-209-0 100 B; 2: 0-68-104-140-176-212-213-177-141-105-69-0 100 B; 3: 0-189-225-261-297-333-332-296-260-224-188-0 100 B; 4: 0-103-102-101-137-136-100-0 100 B; 5: 0-80-79-115-116-117-81-0 100 B; 6: 0-184-220-256-292-328-329-293-257-221-185-0 100 B; 7: 0-76-112-113-114-78-77-0 100 B; 8: 0-45-44-43-42-41-40-0 100 B; 9: 0-53-89-125-161-197-233-269-305-341-340-304-268-232-196-160-124-88-51-0 200 C; 10: 0-72-108-144-179-180-216-252-288-324-360-325-289-253-217-181-145-109-73-0 200 C; 11: 0-60-96-132-168-204-240-276-312-348-349-313-277-241-205-169-133-97-98-0 200 C; 12: 0-50-85-121-157-193-229-265-301-337-336-300-264-228-192-156-120-84-48-0 200 C; 13: 0-56-92-127-128-200-236-272-308-344-345-309-273-237-201-165-129-93-57-0 200 C; 14: 0-49-86-122-158-194-230-266-302-338-339-303-267-231-195-159-123-87-52-0 500 D; 15: 0-1-37-38-39-74-75-111-110-146-182-218-254-290-326-327-291-255-219-183-147-148-149-150-186-222-258-294-330-331-295-259-223-187-151-152-153-154-190-226-262-298-334-335-299-263-227-191-155-119-118-82-83-47-46-0 1500 E; 16: 0-29-28-99-135-134-170-206-242-278-314-350-351-315-279-243-207-171-172-173-174-138-139-175-211-210-246-282-318-354-355-319-283-247-248-284-320-356-357-321-285-249-250-286-322-358-359-323-287-251-215-214-178-142-143-107-106-70-71-0 1500 E; 17: 0-27-26-25-24-23-22-21-20-19-18-54-55-91-90-126-162-198-234-270-306-342-343-307-271-235-199-163-164-166-202-238-274-310-346-347-311-275-239-203-167-131-130-94-95-58-59-61-62-63-64-65-66-67-30-31-32-33-34-35-36-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-0 1900 F

References

[1] Taillard E. A heuristic column generation method for the heterogeneous fleet VRP. RAIRO 1999; 33: 1–34.

[2] Garey M, Johnson D. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: W.H. Freeman; 1979.

[3] Correia I, Gouveia L, Saldanha-da-Gama F. Solving the variable size bin packing problem with discretized formulations. Computers and Operations Research 2008; 35: 2103–13.

[4] Paraskevopoulos D, Repoussis P, Tarantilis C, Ioannou G, Prastacos G. A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. Journal of Heuristics 2008; 14: 425–55.

[5] Brandão J. A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem. European Journal of Operational Research 2009; 195: 716–28.

[6] Li F, Golden B, Wasil E. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. Computers and Operations Research 2007; 34: 2734–42.

[7] Semet F, Taillard E. Solving real-life vehicle routing problems efficiently using tabu search. Annals of Operations Research 1993; 41: 469–88.

[8] Rochat Y, Semet F. A tabu search approach for delivering pet food and flour in Switzerland. Journal of the Operational Research Society 1994; 45: 1233–46.

[9] Brandão J, Mercer A. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. European Journal of Operational Research 1997; 100: 180–91.

[10] Oppen J, Lokketangen A. A tabu search approach for the livestock collection problem. Computers and Operations Research 2008; 35: 3213–29.

[11] Tarantilis C, Kiranoudis C, Vassiliadis V. A list based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. Journal of the Operational Research Society 2003; 54: 65–71.

[12] Tarantilis C, Kiranoudis C, Vassiliadis V. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research 2004; 152: 148–58.

[13] Rochat Y, Taillard E. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1995; 1: 147–76.

[14] Dueck G, Scheuer T. Threshold accepting. A general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 1990; 90(1): 161–75.

[15] Li F, Golden B, Wasil E. Very large-scale vehicle routing: new test problems, algorithms, and results. Computers and Operations Research 2005; 32: 1165–79.

[16] Dueck G. New optimization heuristics: the great deluge algorithm and the record-to-record travel. Journal of Computational Physics 1993; 104: 86–92.

[17] Gendreau M, Hertz A, Laporte G. New insertion and post-optimization procedures for the traveling salesman problem. Operations Research 1992; 40 (6): 1086–94.

[18] Glover F. Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research 1986; 13: 533–49.

[19] Glover F. Tabu search – part I. ORSA Journal on Computing 1989; 1: 190–206.

[20] Glover F. Tabu search – part II. ORSA Journal on Computing 1990; 2: 4–31.

[21] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing. Management Science 1994; 40: 1276–90.

[22] Gendreau M, Laporte G, Musaraganyi C, Taillard E. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. Computers and Operations Research 1999; 26: 1153–73.

[23] Hoff A, Gribkovskaia I, Laporte G, Løkketangen A. Lasso solution strategies for the vehicle routing problem with pickups and deliveries. European Journal of Operational Research 2009; 192:755–66.

[24] Hansen P, Mladenovic N. Variable neighborhood search: principles and applications. European Journal of Operational Research, 2001; 130: 449–67.

[25] Glover F, Laguna, M. Tabu Search. Kluwer Academic Publishers; 1997.

[26] Christofides N, Eilon S. An algorithm for the vehicle-dispatching problem. Operational Research Quarterly 1969; 20:309–18.

[27] Dongarra J. Performance of various computers using standard linear equations software. Report CS-89-85, University of Tennessee; 2006.

[28] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C (Eds). Combinatorial Optimization, Willey, Chichester; 1979: 313-338.

[29] Fisher M. Optimal solution of vehicle routing problems using minimum k-trees. Operations Research 1994; 42 (4): 626-642.