PERGAMON

Computers & Operations Research 26 (1999) 255–270

computers & operations research

A tabu search algorithm for the vehicle routing problem

Gulay Barbarosoglu*,[1], Demet Ozgur[2]

Department of Industrial Engineering, Bogazici University Bebek, Istanbul, Turkey

Received October 1997; received in revised form March 1998

Scope and purpose

This paper deals with the design of a heuristic algorithm to solve the vehicle routing problem (VRP) for a well-known distribution company in Turkey, which transports electronic household commodities from various plants to a large number of dealers. The logistics system between the distribution company and the parent company which owns all the plants is operated on a two-echelon framework; the manufactured commodities are first transported from the plant to the main depot in close proximity of each plant by the plant management and then from the depot to the dealers by the distribution company. Thus, it is just necessary for the distribution company to schedule the delivery from the depots to the dealers. Since each plant manufactures a different commodity and they are located far away from each other, the company management has decided to plan each depot independently. The main issue is to design a fast and robust procedure to solve a single-depot VRP for each plant. Since the VRP is a hard, combinatorial problem, and only relatively small instances can be solved to optimality, an approximate approach is prefered to manage the large number of dealers assigned to each depot. Specifically a tabu search approach known to provide good solutions in the VRP context is employed to obtain a practical solution. The algorithm developed in this study is then tested upon the benchmark problems in literature and shown to provide competitive results. The company management was satisfied with theoretical performance and accepted to use it in their daily operation. The early findings in practice also indicate considerable improvements in operational performance.

* Corresponding author. E-mail: barbaros@boun.edu.tr.

[1] Gülay Barbarosoğlu, Ph.D, is an Associate Professor of Industrial Engineering at Boğaziçi University, İstanbul, Turkey. Her research interests include optimization, hierarchical planning, production planning and control, scheduling, and manufacturing management. She is the Director of Manufacturing Management Systems Laboratory at Boğaziçi University.

[2] Demet Özgür is a doctoral student in the Department of Industrial Engineering at Boğaziçi University, İstanbul, Turkey. She holds a B.S. degree and an M.Sc degree from that university. She is currently conducting doctoral research in optimization of large-scale production, models.

0305-0548/98/$ – see front matter
PII: S0305-0548(98)00047-1

Abstract

The purpose of this study is to develop a new tabu seach heuristic to solve the single-depot vehicle routing problem of a distribution company carrying goods from a depot to a set of dedicated dealers. The heuristic proposes a new neighbourhood generation procedure which considers the scattering pattern in the locations of the dealers. A set of neighbours are generated by first executing a procedure which ignores the clustering of the vertices and then executing a second procedure which takes into account the relative locations of the dealers; then the feasible non-tabu move among these with the best objective function value is implemented. During neighbourhood construction, a combination of traditional improvement techniques are used simultaneously in order to achieve the exchange of vertices. The intensification efforts, on the other hand, are focused upon the hill-climbing behaviour of the search, and the diversification step which is standard in all previous tabu search approaches is not included in this heuristic. Numerical results on well-known benchmark problems indicate that the performance of the algorithm developed in this study is compatible with the other best-known algorithms in the literature.

Keywords: Metaheuristics; Tabu search; Distribution; Vehicle routing problem

1. Introduction

The aim of this study is to introduce a tabu search algorithm for solving the vehicle routing problem (VRP) which can be described as the problem of designing optimal delivery or collection of routes from one or several depots to a number of customers subject to side constraints. The VRP plays a central role in the area of distribution management and logistics, and the costs associated with operating vehicles for delivery purposes form an important component of total logistics costs. The significance of achieving potential savings has motivated the use of analytical techniques in this area of practice, and consequently the theory of VRP has been studied extensively over the past few decades by using different approaches. In fact, the need for developing the VRP algorithm discussed in this paper has arisen in a project dealing with the logistics management problem of a distribution company transporting electronic household commodities manufactured by a parent company in Turkey. The manufacturing company has several plants which are scattered over a large geographical area and have a limited number of depots which they operate themselves. This implies that the transportation from the plants to the depots which are in close proximity of the plants are controlled by the plants and does not pose a logistics problem itself. However, the distribution from the depots to a large number of dealers is operated by the distribution company in a decentralized manner. Since each product is stored in exactly one depot due to the expected savings in material handling and loading requirements, and the aggregate demand requirement of a given product is much bigger than the truck capacity, trucks are always full-loaded with the same product. Although this permits multiple visits to the same demand point by different vehicles, considering the inter-depot distances, demand requirements and truck capacities, no savings can be expected by a multi-depot approach as it was already established by the distribution company management. This implies that each depot may be planned for each product individually and the VRP is to be solved for each one of them separately. Thus, it is decided that a single-depot approach would be sufficient for their purposes.

There exists a wide variety of VRPs and a broad literature on this class of problems. The variants include VRP with time windows, stochastic VRP, multi-depot VRP and others. There have been important advances in the development of exact and approximate algorithms for solving the VRP which is an NP-hard combinatorial optimization problem in nature. As far as the exact approaches are concerned, the most significant progress has been made in the design of branch-and-bound, column generation, Lagrangian decomposition and K-tree algorithms. Laporte and Nobert [1], Laporte, Mercure and Nobert [2], and Fisher [3] are among the ones who have developed various branch-and-bound approaches. Christofides, Hadjiconstantinou and Mingozzi [4] study the dynamic programming approach as applied to the VRP by introducing a state-space relaxation procedure. Very recently, Fisher et al. [5] presented two optimization methods for the VRP with time windows. These are, namely, a generalized K-tree relaxation and a Lagrangian decomposition where variable splitting decomposes the original problem into a semi-assignment problem and a series of shortest path problems. Another Lagrangian approach is introduced by Kohl and Madsen [6] in which the constraint set requiring that each customer be served is relaxed and the subgradient optimization is used to determine the optimal Lagrangian multipliers. Bramel and Simchi-Levi [7] formulate the VRP with time windows as a set covering problem, and first solve the linear programming relaxation of the set covering formulation by using the well-known column generation approach and then implement a branch-and-bound routine to generate an integer solution. It is furthermore shown that as the number of customers increases, the relative gap between the fractional and integer solutions approach zero. As for the approximate algorithms, there exist many studies dating back to Clark and Wright [8], and some among many can be listed as Gillett and Miller [9], Christofides et al. [10], Paessens [11], Desrochers and Verhoog [12], and Altinkemer and Gavish [13]. For an extensive overview of exact and approximate algorithms, the interested reader should refer to Laporte [14]. In this study, the tabu search approach is chosen to solve the VRP in a reasonable amount of computational time with an acceptable degree of precision because the recent research indicates that metaheuristics, especially tabu search, perform well in case of the VRP. One of the first tabu search applications to the VRP is due to Willard [15]. Pureza and França [16] also use the tabu search by swapping vertices between two routes in generating neighbours. Osman [17, 18] uses a combination of 2-Opt moves, vertex assignments to different routes, and vertex interchanges between routes. A novel approach is introduced by Taillard [19] in which the set of vertices are decomposed into subproblems that may be solved independently in order to speed up the iterative search method. Results indicate that it performs better than all the other tabu search approaches. In fact, two different decomposition schemes are proposed for the uniform and nonuniform problems. A decomposition into polar regions is shown to perform well in case of uniform problems where the depot is almost centered and the vertices are regularly distributed around the depot. Vertices and tours are exchanged between the subproblems after a summary resolution of the subproblems. However, a decomposition method based on the partition of the arborescence of the shortest paths from the depots to all the vertices is recommended for the nonuniform problems where the vertices are not regularly distributed around the depot. Here vertices and tours are not transmitted between the resolutions. In all these algorithms, a feasible solution is not permitted to become infeasible while Gendreau et al. [20] permit infeasible moves with respect to the side constraints. The algorithm developed in this study, namely DETABA, uses most of the tabu search principles developed previously, but proposes a new neighbour search procedure without any diversification and a new intensification scheme, and its performance is compared

against the above-mentioned tabu search algorithms using the well-known benchmark problems. It is noted that DETABA in general performs better than all the algorithms except that of Taillard [19]. The paper is organized as follows: The VRP is described and a generic VRP formulation is given in Section 2. The tabu search technique DETABA is discussed in Section 3 while the computational results are provided in Section 4. Some concluding remarks are made in Section 5.

2. The vehicle routing problem

The VRP can be simply stated as the problem of determining optimal routes through a set of locations and defined on a directed graph $G = (V, A)$ where $V = (v_0, v_1, \ldots, v_n)$ is a vertex set and $A = ((v_i, v_j): v_i, v_j \in V, i \neq j)$ is an arc set. Vertex v_0 represents a depot where a fleet of N_v vehicles of the same capacity are located. The value of N_v can be either prespecified or free, i.e. bounded above by a constant $N \leqslant n - 1$. All remaining vertices represent customers. A non-negative (distance/time/cost) matrix $C = (c_{ij})$ is defined on A. Here since $c_{ij} = c_{ji}$ for all (v_i, v_j), the problem is said to be symmetric, and arcs are represented by undirected edges. A non-negative weight d_i is associated with each vertex to represent the customer demand at v_i, and naturally the weight assigned to any route may not exceed the vehicle capacity Q_v. In some problems, a further limitation is imposed on the total route duration in addition to the vehicle capacity constraint. In such a case, t_{ij} is defined to represent the travel time for each (v_i, v_j), t_i represents the service time at any vertex v_i and it is required that the total duration of any route should not exceed a preset bound T_v. Thus, the single-depot VRP aims at determining N_v vehicle routes of minimal total cost, each starting and ending at the depot, so that every customer is visited exactly once, subject to the above-mentioned constraints. A typical mathematical formulation for the single depot VRP is provided below:

$$\text{Minimize} \quad \sum_i \sum_j \sum_v c_{ij} X_{ij}^v \tag{1}$$

$$\text{subject to} \quad \sum_i \sum_v X_{ij}^v = 1 \quad \text{for all } j \tag{2}$$

$$\sum_j \sum_v X_{ij}^v = 1 \quad \text{for all } i \tag{3}$$

$$\sum_i X_{ip}^v - \sum_j X_{pj}^v = 0 \quad \text{for all } p, v \tag{4}$$

$$\sum_i d_i \left(\sum_j X_{ij}^v \right) \leqslant Q_v \quad \text{for all } v \tag{5}$$

$$\sum_i t_i^v \sum_j X_{ij}^v + \sum_i \sum_j t_{ij}^v X_{ij}^v \leqslant T_v \quad \text{for all } v \tag{6}$$

$$\sum_{j=1}^{n} X_{0j}^v \leqslant 1 \quad \text{for all } v \tag{7}$$

$$\sum_{i=1}^{n} X_{i0}^v \leqslant 1 \quad \text{for all } v \tag{8}$$

$$X_{ij}^v \in Z \quad \text{for all } i, j, \text{ and } v \tag{9}$$

In this formulation the decision variables X_{ij}^v are binary variables indicating if arc(v_i, v_j) is traversed by vehicle v. The objective function of distance/cost/time minimization is expressed by Eq. (1). In this study, the locations of the vertices are specified by their respective coordinates and Euclidean distances are used without loss of generality. Constraints (2) and (3) together state that each demand vertex be served by exactly one vehicle. It is guaranteed in Eq. (4) that a vehicle leaves the demand vertex it has already entered. Vehicle capacity constraints are expressed by Eq. (5) whereas the limitation on the maximum route duration is given by Eq. (6). Constraints (7) and (8) express that vehicle availability not be exceeded. The subtour elimination constraints are given in Eq. (9) where Z can be defined by

$$Z = \{(X_{ij}^v): \sum_{i \in B} \sum_{j \in B} X_{ij}^v \leqslant |B| - 1 \quad \text{for } B \subseteq V/\{0\};\ |B| \geqslant 2\}.$$

Since the distribution company under consideration does not limit the route duration in their current management system because of the completely decentralized nature, constraints (6) are not explicitly included in DETABA. However, the definition of the feasible space can be extended easily without loss of generality. In fact, a possible extension of the proposed approach will be provided in Section 3.

As it is indicated before, major advances have taken place in the area of metaheuristics, including simulated annealing, tabu search, genetic search and neural networks; thus, in this project, a metaheuristic approach is chosen to solve the single-depot VRP of the particular distribution company, and the details of this tabu search are provided in the next section.

3. Tabu search algorithm

Tabu search is a metastrategy iterative procedure for building extended neighbourhood with particular emphasis on avoiding being caught in a local optimum. This global optimization metaheuristic was initially proposed by Glover [21], evolved in Glover [22, 23], and Glover et al. [24]. It consists of exploring the search space by moving from a solution to its best neighbour even if this results in a deterioration of the objective function value. This way the likelihood of moving out of local optima is increased. The successive neighbours of a solution are generated and their objective function values are examined. To avoid cycling, solutions that were recently examined are forbidden or declared *tabu* for a certain number of iterations. A move made in iteration t is called *tabu* until iteration $(t + \theta)$ where θ is the tabu duration randomly chosen on a prespecified interval. Thus, the tabu list is an ordered queue containing forbidden moves; whenever a move is made, it is inserted to the end of the tabu list and the first element from the list is removed. The best admissible move is then chosen as the highest evaluation move in the neighbourhood of the current solution in terms of the objective function value and the tabu restrictions. In some studies the whole neighbourhood is explored and the best non-tabu move is selected; however, in some other studies the first feasible non-tabu improving move is selected. Since tabu search permits the disimproving moves as well, it is necessary to restrain the search only to non-tabu moves. An improving move is not accepted if it is forbidden in the tabu list unless it satisfies the *aspiration criterion*. The aspiration criterion is a measure solely designed to override the tabu status of a move if this move

leads to a solution better than the best found by the search so far. *Intensification* strategies can be applied to accentuate the search in a promising region of the solution space, so that the moves to the local optimum are intensified. *Diversification*, on the other hand, can be used to broaden the search into less explored regions by forcing the moves out of the local optimum.

All these concepts are extensively used in the tabu search DETABA developed in this project. Generally in the VRP context, it is a common practice to define a neighbour by either the exchange of vertices between different routes, the replacement of vertices into a different route or within its own route, the creation of a new route or the deletion of an existing route. In this study two different procedures are developed in order to define the neighbours. The first procedure basically does not consider the scatter of the vertices; however, it is observed that some of the dealers are found to be clustered around the large cities, especially in the eastern part of the country, and the algorithm is shown to perform rather poorly in case there exists a clustering pattern in the locations of the vertices. Then a second procedure which takes into account the underlying clustering is developed and used jointly with the first one to define the best move. The neighbourhood construction heuristics called TANE and TANEC, respectively, which attempt to improve upon a given solution are naturally the main heuristics in the tabu search DETABA and both include a combination of 2-Opt moves, vertex reassignments to different routes and vertex interchanges between two routes. First, TANE is executed to explore a given number of neighbours around the current solution and the best non-tabu move is determined. Then TANEC is used to find the best TANEC non-tabu move, and the one with the smaller objective function value is implemented as the final move of the iteration. TANE and TANEC are both characterized by a vector of parameters

$$(N_{\max}, \alpha_{\max}, \beta_{\max}, M, P, \theta_{\min}, \theta_{\max}),$$

where

$N_{\max}$	maximum number of iterations without any improvement
$\alpha_{\max}$	maximum number of neighbours to be generated in the neighbourhood
$\beta_{\max}$	maximum number of 2-Opt exchanges
(M, P)	maximum number of vertices to interchanged between the routes
$(\theta_{\min}, \theta_{\max})$	bounds on the tabu duration

First, the algorithm TANE can be summarized as follows:

Step 1: Execute the following steps to generate a neighbour.
- *Step* 1.1: Randomly choose a route and call it R1
- *Step* 1.2: Randomly choose another route and call it R2
- *Step* 1.3: Randomly choose $\mu \leqslant \min(M, |\mathrm{R1}|)$ vertices out of R1.
- *Step* 1.4: Randomly choose $\pi \leqslant \min(P, |\mathrm{R2}|)$ vertices out of R2.
- *Step* 1.5: If this mutual insertion leads to infeasibility in Eq. (5), then repeat Steps 1.3 and 1.4 to generate a new set of μ vertices out of R1 and a new set of π vertices out of R2.
- *Step* 1.6: Insert the chosen π vertices into R1 by using the insertion procedure.
- *Step* 1.7: Insert the chosen μ vertices into R2 by using the insertion procedure.
- *Step* 1.8: Apply the 2-Opt procedure to R1 and R2 independently for $\beta_{\max}$ times.
- *Step* 1.9: Compute the objective function value of the final situation.

Step 2: Repeat the above steps for $\alpha_{\max}$ times.
Step 3: Implement the non-tabu move with the best objective function value.

As it is indicated before, in case of clustering, the relative proximity of the vertices should be taken into account, and the neighbourhood construction should be modified to account for this. The main modification is made in Step 1 in defining a feasible neighbour and the new procedure with the modified Step 1 is called TANEC and is given below:

Step 1.1: Randomly choose a route and call it R1
Step 1.2: Randomly choose another route and call it R2
Step 1.3: Randomly define $\mu \leqslant \min(M, |\mathrm{R1}|)$.
Step 1.4: Randomly define $\pi \leqslant \min(P, |\mathrm{R2}|)$.
Step 1.5: Start with R1.
1.5.1. Find the centroid of R2 including all vertices. Call this C2
1.5.2. For each $i \in$ R1, find the centroid of R1 excluding v_i. Call this $\mathrm{C1}_i$.
1.5.3. Find the distance between $\mathrm{C1}_i$ and v_i. Call this d1.
1.5.4. Find the distance between C2 and v_i. Call this d2.
1.5.5. Find the ratio d1/d2.
1.5.6. Sort the vertices in decreasing order of {d1/d2}
Step 1.6: Repeat Step 1.5 for R2.
Step 1.7: Choose μ vertices from the ordered array of R1.
Step 1.8: Choose π vertices from the ordered array of R2
Step 1.9: Insert the chosen π vertices into R1 by using the insertion procedure.
Step 1.10: Insert the chosen μ vertices into R2 by using the insertion procedure.
Step 1.11: When the chosen combination of (μ, π) vertices cannot be implemented due to infeasibility in Eq. (5), carry out the search over the ordered arrays favoring the vertices scattered away from the centroid.
Step 1.12: Apply the 2-Opt procedure to R1 and R2 independently for $\beta_{\max}$ times.
Step 1.13: Compute the objective function value of the final situation.

The main idea here is to interchange the vertices which are relatively away from the centroid of their current route and which are relatively close to the centroid of the route into which they are about to be moved. In order to achieve this, for each vertex the centroid of the current route is computed excluding it, and the Euclidean distance is computed between this centroid and its location. Also the distance between the centroid of the next route and its current location is computed, and the ratio between these two distances is found as a measure of relative aptness to be interchanged. The overall search algorithm called SENE can be summarized as follows:

Step 1: Generate the best non-tabu move tentatively by using the procedure TANE.
Step 2: Generate the best non-tabu move tentatively by using the procedure TANEC.
Step 3: The one with the smaller objective function value is implemented as the best non-tabu move.

In DETABA, the tabu restriction is defined so as to avoid the reassignment of previously moved vertices to the original routes they have already moved from. If at iteration k vertices $v_1 \in$ R1 and $v_2 \in$ R2 are exchanged (i.e. v_1: R1 $\rightarrow$ R2 and v_2: R2 $\rightarrow$ R1), it is forbidden to put both v_1 in R1 and v_2 in R2 again during iterations $(k+1, \ldots, k+\theta)$ where θ is the tabu duration chosen randomly

from ($\theta_{\min}$, $\theta_{\max}$). If all the moves in the whole neighbourhood becomes tabu and none satisfies the aspiration criterion, the algorithm chooses the oldest tabu move and proceeds accordingly. In the solution improvement phase periodically, in fact every $5n$ moves, the algorithm DETABA checks if a new best solution has been found. If not, it goes back to the last best solution.

The other key issues to be specified in DETABA are the methods used to construct the initial solutions, the insertion procedure, the 2-Opt improvement procedure and the intensification strategy, each of which will be discussed in detail.

3.1. Construction of an initial solution

It is necessary to generate a number of solutions to initialize the main search process in DETABA. Here two methods are proposed to construct these in a rather random manner without much emphasis on the optimality. Each method will be executed $\sqrt{n}/2$ times generating a total of $\sqrt{n}$ solutions. The best among them will be chosen as the initial solution to the main body of the search itself and will be improved by the main algorithm.

Method 1

Step 1. Generate a random vertex (v_i) and form the vertex sequence

$$(v_0, v_i, v_{i+1}, \ldots, v_n, v_1, v_2, \ldots, v_{i-1})$$

Step 2. Starting with v_0, create routes by following the above vertex sequence such that the first vehicle contains all cities starting from the first city in the sequence $\leqslant Q_v$, and repeat this process until all vertices have been included into the routes.

Method 2

Step 1: Generate a subtour consisting of only vertex v_i chosen randomly.
Step 2: Find v_k such that c_{ik} is minimal and form the subtour (v_0, v_i, v_k, v_0)
Step 3: Find some v_k not already in the subtour closest to any node in the subtour which will not violate Q_v. If there is no such vertex, generate another route by another random vertex chosen from among the remaining ones.
Step 4: Find the arc (v_i, v_j) in the subtour which minimizes ($c_{ik} + c_{kj} - c_{ij}$), and insert v_k between v_j and v_i.

3.2. The insertion procedure

In the neighbour search procedures TANE and TANEC, it becomes necessary to insert several vertices into a given route simultaneously; thus, a simple insertion rule is required to place $\{v_i: v_i \in \mathrm{VP}\}$ into a specific route R where VP is a proper subset of V.

Step 1: Find the vertex $v_k \in \mathrm{VP}$ with minimum c_{ik} value to any vertex already in the route R.
Step 2: Find the arc (i, j) in the route R which minimizes ($c_{ik} + c_{kj} - c_{ij}$) and insert vertex k between i and j.
Step 3: Repeat Steps 1 and 2 until all the vertices in VR are placed in the route R.

3.3. The 2-Opt improvement procedure

A 2-Opt procedure is employed in Step 1.8 of TANE and Step 1.12 of TANEC in order to intensify the solution improvement in each route locally. Suppose a subtour consists of the following set of S vertices in the given order $S = \{v_0, v_1, v_2, \ldots, v_k, v_0\}$, and let $X = \{(v_i, v_{i+1}); (v_j, v_{j+1})\}$ be a set of two edges in S which are to be replaced with edges $Y = \{(v_i, v_j); (v_{i+1}, v_{j+1})\}$ if this replacement will lead to an improvement. Here it is required that all the vertices under consideration be different from each other. Once the set X has been chosen, the set Y is directly determined. In a subtour consisting of k customer vertices and a depot, there are $[(k+1)(k-2)/2]$ possible edge combinations given by the set E. The 2-Opt algorithm can be summarized as follows:

Step 1: For each $X \in E$, calculate the improvement δ_x obtained by replacing X by the associated Y and given by

$$\delta_x = (c_{i,i+1} + c_{j,j+1}) - (c_{ij} + c_{i+1,j+1})$$

Step 2: Calculate $\delta_{\max}$ by

$$\delta_{\max} = \max \{\delta_x\}$$

Step 3: If $\delta_{\max} > 0$, replace the two edges asociated with $\delta_{\max}$ and repeat this for all X in E.

Since limitation upon travel times is not included in DETABA, it has been sufficient to check the feasibility of a move only with respect to constraints (5) in Step 1.5 of TANE and Step 1.11 of TANEC before any vertex insertion or 2-Opt exchanges are actually tried: The demand of vertex i and, consequently, its effect upon the vehicle capacity constraint is naturally independent of its location in the route. However, the inclusion of constraints (6) would require some changes since the total travel time of any vehicle clearly depends upon the sequence of vertices visited in the route. Although different feasibility controls can be incorporated at various points in the overall procedure, it is recommended to guarantee feasibility of constraints (6) within the 2-Opt procedure without changing the insertion procedures since the feasibility state of a possible route can only be justified at the end of the 2-Opt procedure. This implies that starting from a feasible solution the insertion procedures may lead to travel-time infeasibility to be improved by the 2-Opt procedure. Thus, the 2-Opt procedure would be implemented to generate a feasible solution which could preferably improve the objective function further, and $\delta_{\max}$ is determined by considering only the feasible routes. If on the other hand, a feasible solution cannot be obtained at the end of $\beta_{\max}$ exchanges, Steps 1.3 and 1.4 of TANE and Steps 1.7 and 1.8 of TANEC would be repeated to generate new sets of μ and π vertices, respectively. If that attempt fails, two new routes are generated. With travel time restriction, the construction of initial solutions should also be revised to maintain feasibility such that the vertices can be included into the routes as long as the inclusion does not violate Q_v and T_v. Moreover, a preliminary check could be incorporated before the insertion procedures both in TANE and TANEC in order to detect infeasible combinations and accordingly to save redundant 2-Opt trials by just considering the best-case behaviour. The best-case behaviour can be computed as a lower bound on travel time requirement by using t_i^v and the minimum of t_{ij}^v values among all vertices to be newly inserted into a given route and the vertices already remaining in the route without considering any route construction.

3.3.1. Intensification strategy

Since the search methods possess the risk of being trapped in a local minimum and the possibility of overlooking a global optimum hidden in a deep valley of the objective function, the intensification efforts in DETABA are mainly directed as an attempt to characterize the unexplored regions. To do so, during the execution of the solution improvement stage, the solutions which are suspected to indicate such a search direction are identified as good candidates to initialize the intensification stage and are included in the set INS. At the end of the solution improvement step, a given number of solutions with the minimum objective function values among those are selected for intensification. Suppose in iteration k a solution S_k with an objective function value $C(S_k)$ is obtained, and in iteration $k+1$ S_{k+1} is found to be the best admissible move in the neighbourhood generated around the solution S_k, so that the search will move to S_{k+1}. The change in the objective function values, $\Delta_k = C(S_{k+1}) - C(S_k)$, is calculated. If $\Delta_k \leqslant 0$, then it is obvious that the search is improving towards a local minimum; however, if $\Delta_k > 0$, then the search is hill-climbing. This implies that keeping track of Δ_k will provide valuable information as to which solution is worth the intensification effort. Thus if $\Delta_{k-1} \leqslant 0$ and $\Delta_k > 0$, then S_k is identified as the indicator of a promising region which may be worth further exploration, and is included in the set INS. The solutions in INS are stored in increasing order of their objective function values, so that the ones at the top of the list are the most promising ones.

It is important to note that diversification is not explicitly attempted in this algorithm, so that there is no diversification step which is common in all the previous tabu search algorithms. Another important issue is to determine the number of neighbours to be evaluated in each iteration by considering the tradeoff between computational burden and the analytical refinement. It should be naturally a function of the number of vertices as well as the number of vehicles; thus, a lower bound on the number of vehicles $m = \sum d_i/Q$ is computed and used in the determination of $\alpha_{\max}$ as a measure of the minimum number of routes that can be obtained in any iteration. It is also included in the computation of the number of solutions to be improved in the intensification stage. At this point it is possible to summarize the modules of the main tabu search DETABA:

Step 0 (*Initialization*): Generate $\sqrt{n}/2$ initial solutions by using each of the methods discussed in the initial solution generation procedure.

Step 1 (*First Improvement*): Apply the search algorithm SENE to each of the initial solutions with $N_{\max} = n$, $\alpha_{\max} = 5$, $\beta_{\max} = \text{all}$, $M = 2$, $P = 2$, $\theta_{\min} = 5$, $\theta_{\max} = 10$.

Step 2 (*Solution Improvement*): Starting with the best solution obtained in Step 1, apply the search algorithm SENE with $N_{\max} = 50n$, $\alpha_{\max} = n.m/2$, $\beta_{\max} = \text{all}$, $M = 2$, $P = 2$, $\theta_{\min} = 10$, $\theta_{\max} = 20$.

Step 3 (*Intensification*): Apply the search algorithm SENE to the first $\sqrt{n.m}$ solutions with the minimum objective function value from among the solutions determined in Step 2 and included in the set INS with $N_{\max} = n$, $\alpha_{\max} = n.m/2$, $\beta_{\max} = \text{all}$, $M = 2$, $P = 2$, $\theta_{\min} = 5$, $\theta_{\max} = 10$.

4. Computational results

In order to justify the performance of the approach to the company's management, DETABA was tested on the 7 test problems (problems 1–5 and 11–12) given in [10]. These problems contain between 50 and 199 vertices as well as the depot. Without loss of generality, the selected problems have only capacity restrictions, but do not have route length constraints because of the project requirements. However, the experiments can be directly extended to the latter case by modifying the definition of a feasible move. In the problems 1–5, the locations of the vertices are randomly generated over the plane while in the problems 11–12 vertices appear in clusters. Euclidean distances are used in all these cases. The travel time matrix coincides with the distance matrix, and there are no service times at the vertices. All computations were performed with full real precision and the solutions were reported as rounded up or down after the second decimal. The results are provided in Table 1 in terms of the objective function values after the first improvement, solution improvement and intensification phases.

Furthermore, comparisons were made between DETABA and the other heuristics for which computational results are already available in the literature. It is realized that all classical algorithms are clearly dominated by DETABA as it is generally the situation when tabu search algorithms are applied to VRP. The fact that metaheuristics such as simulated annealing and tabu search performs far better in VRP than the classical approximate approaches due to their extensive search process is discussed in [20]. Thus only the solutions for the tabu search algorithms developed in [15–20] are reported in Table 2.

As it can be seen, the best-known solutions are always obtained by Taillard's tabu search in [19]. However DETABA is comparable with all the other algorithms mentioned above. For Problem 1, DETABA obtains the optimal solution $C^* = 524.61$ which is found to be optimal in Hadjiconstantinou and Christofides [25]. In problems 2 and 4, TABUROUTE performs better than DETABA while the reverse is true for Problems 3, 5, and 11. With the exception of Problem 4, DETABA is better than the other tabu search algorithms. Thus, the performance of DETABA was found very satisfactory by the company management and implemented in their decision making system. More specifically the best solution results are also given in Table 3 providing the number of vertices, the

Table 1
Objective function values in each DETABA phase for the test problems

Problem	n	First improvement best solution	Solution improvement best solution	Intensification best solution
1	50	647.60	524.61	524.61
2	75	993.92	839.07	836.71
3	100	973.76	834.02	828.72
4	150	1322.50	1045.10	1043.89
5	199	1745.06	1309.48	1306.16
11	120	1207.44	1051.87	1051.18
12	100	1004.74	820.23	819.56

vehicle capacity, the load in each route, the sequence of vertices visited, and the route duration for all the problems. A thorough analysis reveals that the number of routes generated by DETABA is exactly equal to the number of routes generated by TABUROUTE reported in [20] for all problems.

Table 2
Computational comparison of the best solutions obtained by TS algorithms

Problem	Willard	Pureza/ França	Osman (BA)	Osman (FBA)	TABU ROUTE	Taillard	DETABA
1	588	536	524.61	524.61	524.61	524.61	524.61
2	893	842	844	844	835.77	835.26	836.71
3	906	851	835	838	829.45	826.14	828.72
4	—	1081	1052	1044.35	1036.16	1028.42	1043.89
5	—	—	1354	1334.55	1322.65	1298.79	1306.16
11	—	1049	1042.11	1043	1073.47	1042.11	1051.18
12	—	829	819.59	819.59	819.56	819.56	819.56

Table 3
Best solution results for the test problems

Problem # 1: $n = 50$, $Q = 8000$			Total duration: 524.61
Route	Sequence of vertices	Route load	Route duration
1	0 38 9 30 34 50 16 21 29 2 11 0	7950	99.33
2	0 18 13 41 40 19 42 17 4 47 0	7850	109.06
3	0 32 1 22 20 35 36 3 28 31 26 8 0	7450	118.52
4	0 27 48 23 7 43 24 25 14 6 0	7600	98.45
5	0 12 37 44 15 45 33 39 10 49 5 46 0	8000	99.25
Problem # 2: $n = 75$, $Q = 7000$			Total duration: 836.71
Route #	Sequence of vertices	Route load	Route duration
1	0 12 72 39 9 32 44 3 17 0	6900	65.62
2	0 68 2 28 61 21 74 30 0	7000	74.38
3	0 8 54 19 59 14 53 7 0	6950	95.33
4	0 35 11 66 65 38 58 0	6900	81.81
5	0 16 49 24 56 23 63 33 6 0	7000	92.69
6	0 45 5 15 57 13 27 52 46 0	6950	73.55
7	0 51 73 1 43 41 42 64 22 62 0	6900	100.01
8	0 75 4 34 67 26 0	5850	32.58
9	0 10 31 25 55 18 50 40 0	7000	117.45
10	0 48 47 36 69 71 60 70 20 37 29 0	6750	103.30

Table 3. Continued

Problem # 3: $n = 100$, $Q = 10000$		Total duration:	828.72
Route #	Sequence of vertices	Route load	Route duration
1	0 41 22 23 67 39 56 75 74 72 73 21 40 0	9700	108.65
2	0 50 33 81 51 9 71 65 35 34 78 79 3 77 76 0	9950	118.79
3	0 6 96 99 59 93 85 61 17 45 84 5 60 89 0	9800	82.73
4	0 28 12 80 68 29 24 54 55 25 4 26 53 0	8250	98.25
5	0 92 98 37 100 91 16 86 38 44 14 42 43 15 57 2 58 0	9900	126.66
6	0 52 7 82 48 19 11 64 49 36 47 46 8 83 18 0	9950	138.79
7	0 31 88 62 10 63 90 32 66 20 30 70 1 69 27 0	9950	113.93
8	0 13 87 97 95 94 0	5400	40.91
Problem # 4: $n = 150$, $Q = 200$		Total duration:	1043.89
Route #	Sequence of vertices	Route load	Route duration
1	0 144 57 15 43 38 140 86 113 17 84 5 118 60 0	198	114.84
2	0 53 58 2 115 145 41 22 133 75 74 72 73 21 105 0	197	70.74
3	0 40 110 4 56 23 67 39 139 25 55 130 149 26 0	200	110.25
4	0 88 148 62 123 19 107 11 64 49 143 36 47 7 0	196	120.69
5	0 3 78 34 135 35 136 65 71 66 103 9 120 51 122 0	199	134.63
6	0 54 134 24 29 121 129 79 81 33 102 50 1 132 0	195	89.17
7	0 112 94 13 0	79	27.72
8	0 89 147 6 96 104 99 93 85 98 92 59 95 0	195	53.72
9	0 117 97 37 100 91 61 16 141 44 119 14 142 42 87 137 0	200	86.46
10	0 18 83 114 8 125 45 46 124 48 82 106 52 146 0	189	89.95
11	0 69 101 70 30 20 128 131 32 90 63 126 108 10 31 127 0	197	89.89
12	0 27 111 76 116 77 68 80 150 109 12 138 28 0	190	55.82
Problem # 5: $n = 199$, $Q = 200$		Total duration:	1306.16
Route #	Sequence of vertices	Route load	Route duration
1	0 72 197 56 186 23 75 133 41 22 74 171 73 0	199	57.07
2	0 105 12 109 177 80 150 68 116 196 76 184 0	194	55.34
3	0 18 114 8 174 45 125 199 83 60 118 166 0	192	66.91
4	0 70 30 128 160 131 32 181 63 126 90 108 10 189 0	200	97.09
5	0 53 152 58 40 180 21 198 149 26 138 154 0	200	51.95
6	0 5 84 173 17 113 86 16 61 85 93 6 0	197	84.63
7	0 89 147 96 104 99 59 95 94 183 112 0	200	42.46
8	0 27 167 127 190 31 162 101 69 132 1 176 111 0	199	53.04
9	0 82 46 124 168 47 36 143 49 64 11 175 107 19 123 0	200	130.76
10	0 179 130 165 55 25 170 67 39 187 139 155 4 110 0	199	96.16
11	0 98 193 91 191 141 44 140 38 14 119 192 100 37 151 92 0	197	91.70
12	0 50 102 157 33 81 9 120 164 34 78 185 0	197	81.50
13	0 195 54 134 163 24 29 121 169 129 79 158 3 77 28 0	200	80.02
14	0 13 117 97 87 144 172 42 142 43 15 57 145 115 178 2 137 0	197	92.78
15	0 122 20 188 66 65 136 35 135 71 161 103 51 0	195	116.52
16	0 146 88 148 159 62 182 48 7 194 106 153 52 0	200	73.43
17	0 156 0	19	4.47

Table 3. Continued

Problem # 11: $n = 120$, $Q = 200$			Total duration: 1051.18
Route #	Sequence of vertices	Route load	Route duration
1	0 118 8 12 13 14 15 11 10 9 7 6 5 4 3 1 2 0	199	135.50
2	0 109 21 26 29 32 35 36 34 33 30 31 28 27 24 22 25 23 20 19 16 17 0	197	213.42
3	0 37 38 39 42 41 44 46 49 47 48 50 51 45 43 40 110 0	198	201.60
4	0 100 53 55 58 56 60 63 66 64 62 61 65 59 57 54 52 0	199	213.63
5	0 67 69 70 71 74 75 72 78 80 79 77 76 73 68 98 0	200	145.66
6	0 119 81 112 84 117 113 83 108 114 18 90 91 92 89 85 86 111 82 0	200	67.95
7	0 88 87 95 102 101 99 96 93 94 97 115 116 103 104 107 106 105 120 0	182	73.41
Problem # 12: $n = 100$, $Q = 200$			Total duration: 819.56
Route #	Sequence of vertices	Route load	Route duration
1	0 99 100 97 93 92 94 95 96 98 0	190	95.94
2	0 13 17 18 19 15 16 14 12 10 0	200	96.04
3	0 32 33 31 35 37 38 39 36 34 0	200	97.23
4	0 66 62 74 63 65 67 0	150	43.59
5	0 59 60 58 56 53 54 55 57 0	200	101.88
6	0 81 78 76 71 70 73 77 79 80 72 61 64 68 69 0	200	137.02
7	0 91 89 88 85 84 82 83 86 87 90 0	170	76.07
8	0 43 41 40 42 44 46 45 48 51 50 52 49 47 0	160	65.48
9	0 5 3 7 8 11 9 6 4 2 1 75 0	170	56.17
10	0 21 22 23 26 28 30 29 27 25 24 20 0	170	50.80

Table 4
Computational comparison of the CPU times (s) required by TS algorithms

Problem	Willard	Pureza/ França (1)	Osman (BA) (2)	Osman (FBA) (2)	TABU ROUTE (3)	Taillard (4)	DETABA (5)
1	—	973	67.2	114.0	360	49.0	455.28
2	—	155	70.8	50.3	3228	53.0	2401.62
3	—	1796	675.0	1543.0	1104	580.0	2039.72
4	—	7351	3075.0	3560.0	3528	3800.0	4603.53
5	—	—	1972.7	3246.0	5454	3000.0	7595.05
11	—	1776	1398.4	1445.4	1332	4600.0	4212.41
12	—	3527	407.5	892.2	960	340.0	2277.25

Note: (1) MC68020 microcomputer.
(2) VAX 8600 computer.
(3) Silicon Graphics workstation, 36 MHz, 5.7 Mflops.
(4) Silicon Graphics 4D/35.
(5) Pentium 133 MHz-32MB Ram.

It is observed that the run times are extremely sensitive to search parameters and the seeds of random number generators used in the various procedures of the algorithm. Furthermore, it is well-known that the hardware configurations with different capabilities and the programming facility of parallel computing are important factors affecting the run time; thus, one should be cautious in making equitable comparisons of run times. Here run times across the algorithms are included in Table 4. As it can be seen, generally DETABA takes longer than all the other algorithms basically due to the simultaneous use of TANE and TANEC and two different initial solution generators. Still the improvements realized by these justify higher computation times and the main structure of the approach is not modified. Furthermore in the particular application, it is observed that the average number of vertices varies between 70 and 95, and it approximately takes about 1000 s on the same hardware, which is considered acceptable for a program used weekly.

5. Conclusion

A new tabu search algorithm for the VRP is developed in this study aiming to satisfy the needs of a particular distribution company which plans to use the algorithm repeatedly several times a week. Thus, the emphasis is directed towards developing a methodology which will provide reliable and practical solutions. Although it basically uses the concepts developed in previous tabu search algorithms, it has two original contributions: the intensification stage and the neighbour construction procedure SENE. First of all the intensification is commenced from the solutions which are suspected to bracket a local or hopefully the global optimum. During the course of the solution improvement, the behaviour of the objective function is closely scrutinized, and the solutions which happen to be the first disimprovement after a series of improvements are stored so as to indicate the commencement of hill-climbing around a potential valley of the objective function. The intensification is executed around these solutions to explore those regions more closely. A reasonable number of such solutions are determined by normalizing the number of vertices by a lower bound of the number of routes.

In many tabu search algorithms, the exchange of vertices between different routes is commonly used in the construction of neighbours. In this paper the same concept has been used simultaneously with the 2-Opt exchanges; in fact, the 2-Opt search is employed to bring further local improvement for each route in the solution under consideration. It was decided not to include the diversification stage in the first design to bring some computational savings and to incorporate it later if the initial tests would turn out to be unsatisfactory. However having observed the performance and receiving positive response from the company, it is decided not to include it at all. In short the results indicate that DETABA is compatible with the best tabu search algorithms developed before and can be used in practice with sufficient reliability and speed.

References

[1] Laporte G, Nobert Y. Exact algorithms for the vehicle routing problem. In: Martello S, Laporte G, Minoux M, Riberio C, editors. Surveys in Combinatorial Optimization, Amsterdam: North-Holland, 1987, 147–84.

[2] Laporte G, Mercure H, Nobert Y. A branch-and-bound algorithm for a class of asymmetrical vehicle routing problems. Journal of Operational Research Society 1992;43:469–81.
[3] Fisher ML. Optimal solution of vehicle routing problems and minimum k-trees, Report 89-12-13, Decision Sciences Deparment, The Wharton School, Philadelphia, PA, 1989.
[4] Christofides N, Hadjiconstantinou E, Mingozzi A. A new exact algorithm for the vehicle routing problem based on q-path and k-shortest path relaxations. Working Paper, The Management School, Imperial College, London, 1993.
[5] Fisher ML, Jörnsten KO, Madsen OBG. Vehicle routing with time windows: two optimization algorithms. Operations Research 1997;45:488–92.
[6] Kohl N, Madsen OBG. An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. Operations Research 1997;45:395–406.
[7] Bramel J, Simchi-Levi D. On the effectiveness of set covering formulations for the vehicle routing problem with time windows. Operations Research 1997;45:295–301.
[8] Clarke G, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 1964;12:568–81.
[9] Gillett B, Miller L. A Heuristic algorithm for the vehicle dispatch problem. Operations Research 1974;22:340–49.
[10] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C. editors. Combinatorial optimization, Chichester: Wiley, 1979;315–38.
[11] Paessens H. The savings algorithm for the vehicle routing algorithm. European Journal of Operational Research 1988;34:336–44.
[12] Desrochers M, Verhoog TW. A matching based savings algorithm for the vehicle routing problem. Cahier du GERAD G-89-04, Ecole des Hautes Etudes Commerciales de Montreal, 1989.
[13] Altinkemer K, Gavish B. Parallel savings based heuristic for the delivery problem. Operations Research 1991;39:456–69.
[14] Laporte G. The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research 1992;59:345–58.
[15] Willard JAG. Vehicle routing using r-optimal tabu search, M.Sc. Dissertation, The Management School, Imperial College, London, 1989.
[16] Pureza VM, França PM. Vehicle routing problems via tabu search metaheuristic, Publication CRT-747, Centre de recherche sur les transports, Montreal, 1991.
[17] Osman IH. Metastrategy simulated annealing and tabu search algorithms for combinatorial optimization problems. Ph.D. Dissertation, The Management School, Imperial College, London, 1991.
[18] Osman IH. Metastrategy simulated annealing and tabu search algorithms for combinatorial optimization problems. Annals of Operations Research, 1993;41:421–51.
[19] Taillard E. Parallel iterative search methods for vehicle routing problems. Networks 1993;23:661–72.
[20] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40:1276–90.
[21] Glover F. Heuristic for integer programming using surrogate constraints. Decision Sciences 1977;8:156–66.
[22] Glover F. Tabu search, Part I. ORSA J. Computing 1989;1:190–206.
[23] Glover F. Tabu search, Part II. ORSA J. Computing 1990;2:4–32.
[24] Glover F, Taillard E, de Werra D. A user's guide to tabu search. Annals of Operations Resarch 1993;41:3–28.
[25] Hadjiconstantinou E, Christofides N. An optimal procedure for solving basic vehicle routing problems, presented at the 35th Annual Conf. of the Canadian Operational Research Society, Halifax, Canada, 1993.