Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 595–619
computers & operations research
www.elsevier.com/locate/cor

A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service

Fermín Alfredo Tang Montané, Roberto Diéguez Galvão*

Programa de Engenharia de Produção, COPPE/Federal University of Rio de Janeiro, 21945-970 Rio de Janeiro, RJ, Brazil

Abstract

The vehicle routing problem with simultaneous pick-up and delivery (VRP_SPD) is a variant of the classical vehicle routing problem (VRP) where clients require simultaneous pick-up and delivery service. Deliveries are supplied from a single depot at the beginning of the vehicle's service, while pick-up loads are taken to the same depot at the conclusion of the service. One important characteristic of this problem is that a vehicle's load in any given route is a mix of pick-up and delivery loads.

In this paper we develop a tabu search algorithm to solve VRP_SPD. This algorithm uses three types of movements to obtain inter-route adjacent solutions: the relocation, interchange and crossover movements. A *2-opt* procedure is used to obtain alternative intra-route solutions. Four types of neighbourhoods were implemented, three of them defined by the use of each of the single inter-route movements and the fourth by using a combination of these movements. Two different search strategies were implemented for selecting the next movement: first admissible movement and best admissible movement. Intensification and diversification of the search were achieved through frequency penalization. Computational results are reported for a set of 87 test problems with between 50 and 400 clients.

Keywords: Vehicle routing; Pick-up and delivery service; Tabu search

1. Introduction

One variation of the classical vehicle routing problem considers clients that require simultaneous pick-up and delivery service. We call this problem the vehicle routing problem with simultaneous pickup and

* Corresponding author. Tel.: +55-21-2562-7045; fax: +55-21-2270-9702.
E-mail address: galvao@pep.ufrj.br (R.D. Galvão).

0305-0548/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2004.07.009

delivery (VRP_SPD). VRP_SPD was introduced in 1989 by Min [1]. It is often encountered in practice, for example in the soft drink industry, where empty bottles must be returned, and in the delivery to grocery stores, where reusable pallets/containers are used for the transportation of merchandise. In other practical applications within the context of distribution systems, customers may have both a delivery and a pick-up demand. These customers may not accept to be serviced separately for the delivery and pick-up they require, because a handling effort is necessary for both activities and this effort may be considerably reduced by a simultaneous operation.

Reverse logistics is another area in which the planning of vehicle routes takes the form of a VRP_SPD problem, as companies become interested in gaining control over the whole lifecycle of their products. For example, in some countries legislation forces companies to take responsibility for their products during lifetime, especially when environmental issues are involved (as in the disposal of laser printers' cartridges). Returned goods are another example where the definition of vehicle routes may take the form of a VRP_SDP problem.

In Brazil VRP_SPD was studied within the context of transportation of personnel between the continent and oil exploration and production platforms located in the Campos Basin in the state of Rio de Janeiro. In this case, the transportation is carried out by helicopters based on the continent. The problem consists of determining routes for a fleet of helicopters that transports people from the continent to the platforms and back, seeking to minimize transportation costs. A heuristic algorithm was developed to solve this problem, see Galvão and Guimarães [2].

There are few references to VRP_SPD in the literature. There exist, however, abundant references related to routing problems in which clients require pick-up and delivery service, but not simultaneously. There are theoretical relationships between these problems and VRP_SPD and it is possible to transform VRP_SPD into other routing problems with pick-up and delivery service. For example, it is easy to show that any particular instance of VRP_SPD can be transformed into a problem with non-simultaneous pick-ups and deliveries, since the latter problem is a particular case of VRP_SPD in which only one of the demands (pick-up or delivery) is different from zero in each node. It is also possible to show that any instance of VRP_SPD can be transformed into an instance of the *m-Dial-a-Ride Problem*. These transformations are illustrated at the end of Section 2 (see Fig. 2), after several problems involving pick-up and delivery service are defined and briefly reviewed.

This paper is organized in the following manner. In Section 2, we present a brief literature review of vehicle routing problems that involve pick-up and delivery service. In Section 3, two integer programming formulations of VRP_SPD are described (corresponding to problems without and with maximum distance constraints, respectively), and lower bounds are defined for these formulations. The tabu search meta-heuristic is developed in Section 4. This is followed by computational results (Section 5) and conclusions (Section 6).

2. Brief literature review of VRP_SPD and related problems

After VRP_SPD was defined by Min [1] there was, to the best of our knowledge, a gap of more than 10 years without any work on this problem being published. A number of researchers re-visited the problem in recent years. Dethloff [3] studied the problem from the point of view of reverse logistics. He proposed a mathematical formulation for VRP_SPD and developed insertion-based heuristics that use four different criteria to solve the problem. Salhi and Nagy [4] proposed four insertion heuristics, based

on the methodology proposed by Golden et al. [5] and Casco et al. [6]. These same authors recently proposed a local search heuristic that considers solutions with a certain degree of infeasibility (see Salhi and Nagy [7]).

Two local search heuristics were developed by Tang and Galvão [8]. The first of these is an adaptation of a tour partitioning heuristic defined by Beasley [9] and the second uses the sweep algorithm of Gillet and Miller [10], both originally developed for the classical VRP. The procedures developed by Tang and Galvão [8] solve traveling salesman problems with simultaneous pick-up and delivery (TSP_SPD) for each individual route, by adapting methods originally proposed for the non-simultaneous case. Tang and Galvão also proposed an alternative mathematical formulation for VRP_SPD. An exact algorithm for VRP_SPD with time windows was developed by Angelelli and Mansini [11], using a branch-and-price algorithm.

The *Express Delivery Problem* (EDP, see Tang et al. [12]) is another problem with pick-up and delivery demand in each node. In this case, however, the fulfilment of demand is not simultaneous, being composed of two phases: a pick-up phase and a delivery phase. It is worth noting that the pick-up and delivery routes do not necessarily coincide.

Several other routing problems with pick-ups and deliveries are reported in the literature. In these each client requires pick-up or delivery service, but not pick-up and delivery simultaneously. Savelsbergh and Sol [13] define a general routing problem in this category which they call the *General Pick-up and Delivery Problem*. This problem was proposed such that most pick-up and delivery problems can be defined as particular cases of their formulation.

A classical problem addressed in the literature, the *Dial-a-Ride Problem*, consists of picking up clients in pre-specified locations and transporting them to known delivery points, using vehicles based in a given depot. Each pick-up location is associated with a delivery location, forming pairs of locations, with the pick-up activity preceding the delivery activity. This problem may be subject to additional constraints such as maximum travel times for clients and/or time windows. The following objectives are minimized in a hierarchical fashion: (i) number of vehicles; (ii) total distance traveled; (iii) the difference between effective pick-up and delivery times and those desired by clients.

Exact dynamic programming algorithms were developed by Psaraftis [14,15] to solve static and dynamic versions of the *Dial-a-Ride Problem*, as well as a variant with time windows. A more efficient algorithm for the time windows variant was proposed by Desrosiers et al. [16]. Heuristic methods for this problem were developed by Psaraftis [17], Sexton and Bodin [18,19] and Sexton and Choi [20]. More recently Van der Bruggen et al. [21] proposed a simulated annealing algorithm and Madsen et al. [22] used a parallel insertion heuristic for the variant with time windows. The extension of this problem for several vehicles, the *m-Dial-a-Ride problem*, was studied by Roy et al. [23,24], Jaw et al. [25], Desrosiers et al. [26] and Ioachim et al. [27], among others. Dumas et al. [28] proposed an exact procedure to solve the multi-vehicle problem with time windows.

Another routing problem in this category is the problem with *backhauls* (VRP_B); in this problem all deliveries must be concluded before any pick-ups can be made. In the case of a single vehicle the *Traveling Salesman Problem with Backhauls* is defined; Gendreau et al. [29] developed several two-phase heuristics for this problem. The extension for several vehicles was studied by Deif and Bodin [30], Golden et al. [5] and Casco et al. [6]. Toth and Vigo [31] and Mingozzi and Giorgi [32] developed mathematical formulations and exact methods to solve this problem. Gélinas et al. [33] studied the problem with time windows and Duhamel et al. [34] developed a tabu search for solving it.

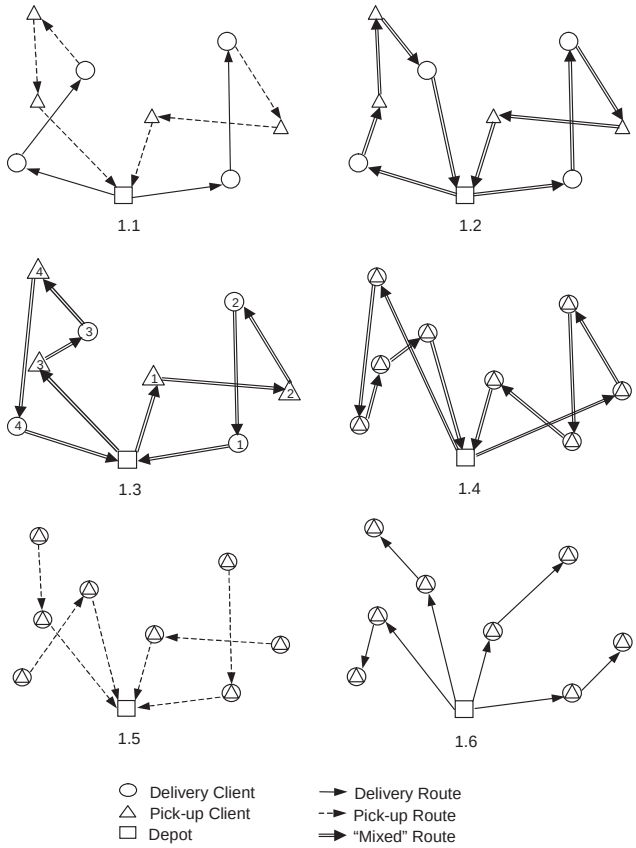

Fig. 1. VRPs with pick-up and delivery service: (1.1)VRP_B; (1.2) VRP_PD; (1.3) *m-Dial-a-Ride Problem*; (1.4) VRP_SPD; (1.5)–(1.6) pick-up and delivery phases of EDP.

A class of problems in which the precedence constraints are relaxed (i.e., it is possible to alternate between pick-ups and deliveries in any given route) has been the object of recent studies. Mosheiov [35] studied the one-vehicle problem. The extension for several vehicles (VRP_PD) was addressed by Mosheiov [36] for the particular case where all pick-up and delivery demands are equal to unity. An illustration of the problems reviewed above is shown in Fig. 1.

Fig. 2 illustrates how VRP_SPD can be transformed into both VRP_PD and the *m-dial-a-ride problem*; for simplicity we show only one route. In Fig. 2.1, we show a route of VRP_SPD with two clients; the numbers shown in parenthesis are, respectively, pick-up and delivery loads. Fig. 2.2 shows the corresponding VRP_PD equivalent: each node is split into a delivery node and a pick-up node, the distance between these two nodes being made equal to zero.

The transformation into the *m-dial-a-ride problem* is shown in Fig. 2.3. Initially, it is necessary to make "dummy" copies of the depot, two copies for each original client; each original node must also be split into two nodes, as in Fig. 2.2. The distances between the depots must be made equal to zero. Clients *1a* and *2a* are collected in "dummy" depots and delivered to their destinations; clients *1b* and *2b* are collected in their origins and delivered to "dummy" depots.

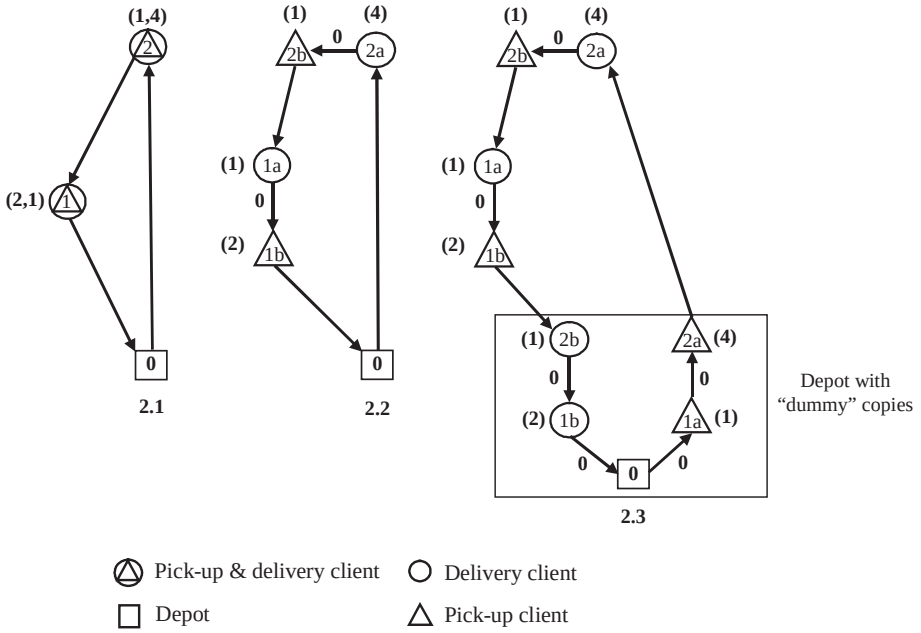

Fig. 2. Equivalence between pick-up and delivery problems.

To the best of our knowledge the algorithm we develop in this paper is the first attempt to solve VRP_SPD using metaheuristics. The mathematical formulation shown in Section 3 is original and the only that we know that considers maximum distance constraints for this problem. Our computational results improve upon previous work and we provide lower bounds that show the quality of our solutions.

3. Mathematical formulations and lower bounds for VRP_SPD

The mathematical formulation proposed by Mosheiov [36] for VRP_PD was extended by us for VRP_SPD. This formulation belongs to a set of formulations based on commodity flows developed for vehicle routing problems (see Gavish and Graves [37]) and does not include maximum distance constraints. Some of the test problems we use to evaluate our algorithm, however, are problems with maximum distance constraints (see Section 5). In order to be able to obtain lower bounds for these problems we developed a second formulation that includes maximum distance constraints.

The formulation without maximum distance constraints is a particular case of the more general formulation that includes maximum distance constraints. In practice it is however possible to use a more compact formulation for the unconstrained case and this is exactly what we do. We only show however the more general formulation. After its presentation a few comments are made about the compact formulation that we use for the unconstrained case.

3.1. A mathematical formulation with maximum distance constraints

Notation

V	set of clients
V_0	set of clients plus depot (client 0): $V_0 = V \cup \{0\}$
n	total number of clients: $n = \|V\|$
c_{ij}	distance between clients i and j
p_j	pick-up demand of client j, $j = 1, \ldots, n$
d_j	delivery demand of client j, $j = 1, \ldots, n$
Q	vehicle capacity
MD	maximum distance allowed for any route k
$\bar{k}$	maximum number of vehicles

Decision variables

$$x_{ij}^k = \begin{cases} 1, & \text{if arc}(i, j) \text{ belongs to the route operated by vehicle } k, \\ 0, & \text{otherwise.} \end{cases}$$

y_{ij}	demand picked-up in clients routed up to node i (including node i) and transported in arc (i, j)
z_{ij}	demand to be delivered to clients routed after node i and transported in arc (i, j)

The corresponding mathematical formulation is given by

$$\text{Minimize} \quad \sum_{k=1}^{\bar{k}} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^k \tag{1}$$

subject to

$$\sum_{i=0}^{n} \sum_{k=1}^{\bar{k}} x_{ij}^k = 1, \quad j = 1, \ldots, n, \tag{2}$$

$$\sum_{i=0}^{n} x_{ij}^k - \sum_{i=0}^{n} x_{ji}^k = 0, \quad \begin{matrix} j = 0, \ldots, n, \\ k = 0, \ldots, \bar{k}, \end{matrix} \tag{3}$$

$$\sum_{j=1}^{n} x_{0j}^k \leqslant 1, \quad k = 1, \ldots, \bar{k}, \tag{4}$$

$$\sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^k \leqslant \text{MD}, \quad k = 1, \ldots, \bar{k}, \tag{5}$$

$$\sum_{i=0}^{n} y_{ji} - \sum_{i=0}^{n} y_{ij} = p_j, \quad \forall j \neq 0, \tag{6}$$

$$\sum_{i=0}^{n} z_{ij} - \sum_{i=0}^{n} z_{ji} = d_j, \quad \forall j \neq 0, \tag{7}$$

$$y_{ij} + z_{ij} \leqslant Q \sum_{k=1}^{\bar{k}} x_{ij}^{k}, \quad i, j = 0, \ldots, n, \tag{8}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 0, \ldots, n, \tag{9}$$

$$y_{ij} \geqslant 0, \quad i, j = 0, \ldots, n, \tag{10}$$

$$z_{ij} \geqslant 0, \quad i, j = 0, \ldots, n. \tag{11}$$

The objective function seeks to minimize total distance traveled. Constraints (2) ensure that each client is visited by exactly one vehicle; constraints (3) guarantee that the same vehicle arrives and departs from each client it serves. Restrictions (4) define that at most $\bar{k}$ vehicles are used; restrictions (5) are the maximum distance constraints. Constraints (6) and (7) are flow equations for pick-up and delivery demands, respectively; they guarantee that both demands are satisfied for each client. Restrictions (8) establish that pick-up and delivery demands will only be transported using arcs included in the solution; they further impose an upper limit on the total load transported by a vehicle in any given section of the route. Finally, constraints (9)–(11) define the nature of the decision variables.

In a broader sense the problem constraints guarantee that each vehicle leaves the depot with a volume equivalent to the sum of the delivery demands of the clients in the route serviced by that vehicle, that each vehicle returns to the depot with a volume equivalent to the sum of the pick-up demands of the clients in the same route, and that the capacity and maximum distance constraints are not violated.

Note that when restrictions (5) are not present this formulation defines the unconstrained problem. In this case, however, it is not necessary to identify each vehicle individually. A more compact formulation, in which only two indices are present, can be used in this case. We use this compact formulation in the calculation of the lower bounds for problems without maximum distance constraints.

3.2. Lower bounds

Two Lagrangean relaxations were defined for the formulation without maximum distance constraints (the compact formulation mentioned above). Neither of the two relaxations produced strong lower bounds. We did not attempt to define relaxations for the formulation with maximum distance constraints.

We obtained lower bounds for the two problems by an alternative methodology, running the models of Section 3 using version 9.0 of CPLEX. Since these are hard combinatorial problems, it is doubtful whether CPLEX can find optimal solutions in reasonable computing times. We adopted therefore the strategy of using CPLEX to run each of the test problems corresponding to the two formulations

for limited amounts of time in a Pentium IV 2.4 GHz. PC with 512 Mb of RAM (see Section 5.1 for detailed results).

4. A tabu search heuristic for VRP_SPD

The state-of-the-art in solution methods for VRPs indicates that metaheuristics, especially tabu search, perform extremely well when compared to other approaches, producing high-quality solutions in reasonable amounts of computing times. Several tabu search algorithms have been proposed in the literature to solve the classical VRPs well as widely studied variants of this problem such as VRPs with time windows and backhauls. It is, however, beyond the scope of the present paper to make a survey of tabu search methods applied to VRPs of different nature.

Tabu search is a procedure that uses an initial solution as a starting basis for seeking improved solutions by searching different neighbourhoods. We start by describing a constructive heuristic that was used to generate this initial solution, proceed discussing issues of interest such as the definition of neighbourhoods, the neighbourhood search, short-term memory (tabu constraints, tabu tenure, aspiration criterion), long-term memory (intensification, diversification), and then finally describe the general structure of a tabu search heuristic for VRP_SPD.

4.1. Initial solution: constructive heuristics for VRP_SPD

As mentioned in Section 2, two heuristic procedures developed for the classical VRP have been extended to solve VRP_SPD by Tang and Galvão [8]. These are adaptations of the *tour partitioning heuristic* of Beasley [9] and of the *"sweep" algorithm* of Gillet and Miller [10].

In the *tour partitioning heuristic* the grouping of clients is made based on a traveling salesman tour, which is partitioned in a sequencial manner. Once the groups of clients are formed, the routing of each group is performed by solving a *Traveling Salesman Problem with Simultaneous Pick-up and Delivery* (TSP_SPD). The tour partitioning is repeated for different starting nodes, with the objective of improving the quality of the solution.

In the algorithm of Gillet and Miller [10] the clients are grouped according to their polar coordinates. The routes are built sequentially and clients may be added or removed form the current route if this reduces the value of the objective function. Once the groups of clients are formed, the routing of each group is also performed by solving a TSP_SPD.

Tang and Galão [8] used four different heuristics to solve TSP_SPD: The *Initial Node Heuristic* and the *Cheapest Feasible Insertion Heuristic*, both proposed by Mosheiov [35], the *Minimum Spanning Tree Heuristic*, proposed by Anily and Mosheiov [38], and the *Cycle Heuristic* of Gendreau et al. [39]. The theoretical basis for the development of these heuristics can be found in these papers.

The heuristic procedures mentioned above were incorporated both to the *tour partitioning heuristic* and to the adaptation of the algorithm of Gillet and Miller developed for VRP_SPD, generating eight constructive heuristic procedures for this problem. The results produced by the tour partitioning heuristics where, on average, of much better quality than those produced by the "sweeping" heuristics. The best results were obtained when the *Cheapest Feasible Insertion* and the *Cycle Heuristic* were used to solve TSP_SPD.

The procedures described above were modified to produce initial solutions for the tabu search heuristic that is the object of the present paper.

4.1.1. Obtaining an initial solution for the tabu search

Grouping strategies: Two strategies were used for grouping the clients, using either upper or lower bounds on the maximum load a vehicle can carry. When upper bounds are used the solution of a TSP for the grouped clients produces a corresponding feasible route. When lower bounds are used a TSP_SPD must be solved to produce a feasible route for the grouped clients.

We define the net demand of a client *i*, *nd*(*i*), as the difference between the pick-up and delivery demands of this client. If $nd\ (i) < 0$ the vehicle load will be reduced by *nd*(*i*) after it visits client *i*; if $nd(i) > 0$ its load will be increased by *nd*(*i*) after this visit. Given a group of clients the *smallest maximum load* occurs when the vehicle visits first all clients with $nd(i) \leqslant 0$ and only then clients with $nd(i) > 0$; the *largest maximum load* occurs when it visits first all clients with $nd(i) > 0$.

The first strategy used to group the clients adds new clients to a group only if the value of the *smallest maximum load* does not exceed vehicle capacity; this strategy does not guarantee a feasible route after the group is formed and a TSP_SPD must be solved to produce the corresponding route. The second strategy adds new clients to a group only if the value of the *largest maximum load* does not exceed vehicle capacity; this strategy guarantees a feasible route and the solution of a TSP produces the desired route.

Routing procedures: The classical strategy group-first, route-second, can be applied to problems without maximum distance constraints. As in this paper we consider problems with and without maximum distance constraints, two different procedures had to be defined to construct initial solutions:

(i) *Independent grouping and routing* (*IGR*), corresponding to the group-first, route second strategy; and
(ii) *Simultaneous grouping and routing* (*SGR*), used when maximum distance constraints are present. In this case, the insertion of clients in a group is subject to two feasibility tests. The first test determines whether there is still capacity in the vehicle to accommodate the client. The second test determines maximum distance feasibility. In this case, the client is only included in the group if its inclusion does not violate either capacity or distance feasibility. In the SGR procedure it is necessary to route the clients to test maximum distance feasibility. The routing of a new client in this procedure uses the principle of *cheapest feasible insertion*.

Heuristic procedures: Four heuristic procedures were implemented for obtaining an initial solution to the tabu search, by combining the two grouping strategies and the two routing procedures described above. We call these heuristics IGR1, IGR2, SGR1 and SGR2, respectively:

- Procedure IGR1—The partitioning of the initial tour is made according to the *smallest maximum load* strategy. In this case, it is necessary to solve a TSP_SPD to obtain a feasible tour for each group of clients. A 2-opt heuristic improves the initial route provided by the TSP_SPD algorithm.
- Procedure IGR2—The partitioning of the initial tour is made according to the *largest maximum load* strategy. The partitioning of the initial tour provides in this case a feasible route for each group of clients; classical TSP heuristics are then used to reduce route lengths.
- Procedure SGR1—The admission of a new client to the group is made according to the *smallest maximum load* strategy. If the new client passes this first feasibility test, it is then routed to determine

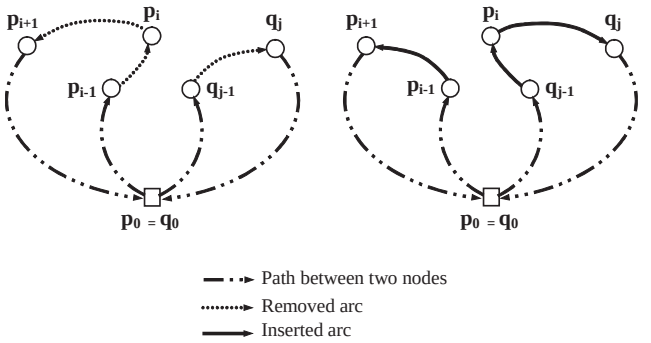

Fig. 3. Neighbourhoods for VRP_SPD: relocation movement.

whether the maximum distance constraint will be satisfied with its inclusion in the group. The *cheapest feasible insertion* principle is used to route the clients.

- Procedure SGR2—Similar to procedure SGR1, the only difference being that admission to a group is made according to the *largest maximum load* strategy.

4.2. Definition of neighbourhoods for the tabu search

We use four types of movement to define neighbourhoods for the tabu search. Three of these are inter-route movements and the fourth is an intra-route movement that improves the routes obtained by the inter-route movements. The inter-route movements are the *Relocation*, *Interchange* and *Crossover* movements; the intra-route movement is a *2-opt* procedure.

Relocation: This movement consists of removing a client from one of the routes and including it in another route, as illustrated in Fig. 3. The movement is in principle attempted for all possible pairs of routes and for each client in each of the two routes. In the example of Fig. 3 client p_i is removed from the first route and inserted into the second.

Interchange: This movement consists of exchanging clients between two routes, as illustrated in Fig. 4. The movement is in principle attempted for all possible pairs of routes and for all possible combinations of exchange of clients between the two routes. In the example of Fig. 4 clients p_i and q_j are exchanged between the two routes.

We call *extended interchange movement* the extension of this movement that corresponds to interchanging two paths. In this case, several clients are transferred between routes, maintaining their order in the original routes.

Crossover: This movement, which is a particular case of the *extended interchange movement* described above, consists of dividing each of two selected routes into two sections. This is achieved by removing an arc from each route and inserting two new arcs that connect, respectively, the initial section of the first route to the final section of the second route and vice versa. This is illustrated in Fig. 5. The movement is in principle attempted for all possible pairs of routes and for all possible partitions of each of the two routes.

2-Opt: This is an intra-route movement whose objective is to improve the solutions obtained by any of the three movements described above. It consists of replacing two non-adjacent arcs belonging to

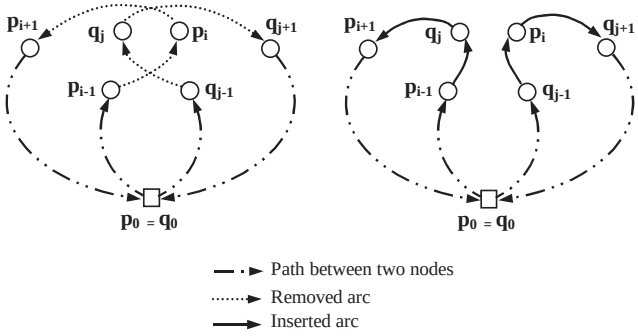

Fig. 4. Neighbourhoods for VRP_SPD: interchange movement.

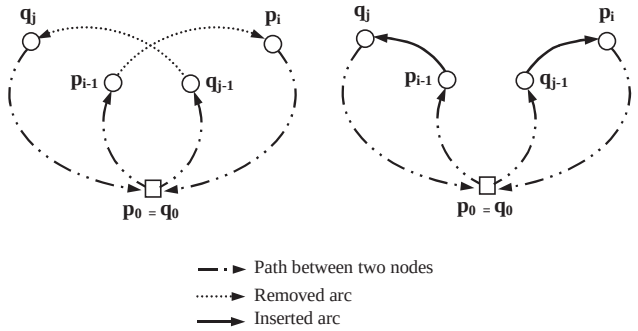

Fig. 5. Neighbourhoods for VRP_SPD: crossover movement.

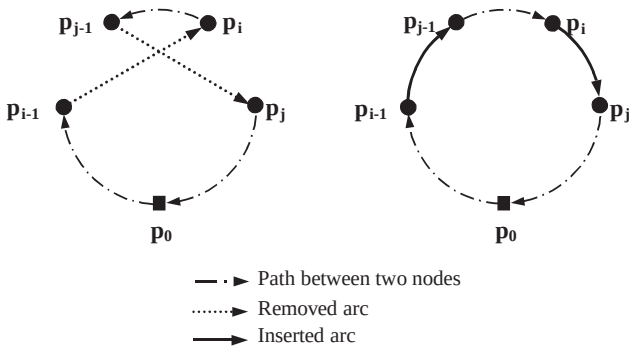

Fig. 6. Neighbourhoods for VRP_SPD: 2-opt movement.

the route under consideration by two other arcs not belonging to it, such that the connectivity of the route is re-established. The process is repeated until no further reduction of route length is possible. This movement is illustrated by the example shown in Fig. 6.

After any of the three inter-route movements described above is executed, the vehicle load must be updated for every client of the two routes involved in the movement. The update process may be described by general equations. These are however omitted.

4.3. The neighbourhood search

Each type of inter-route movement (relocation, interchange, crossover) generates its own neighbourhood. These may be explored in the following ways:

- *First admissible movement*: a neighbourhood is searched until the first feasible movement is found.
- *Best admissible movement*: a neighbourhood is totally searched; the best feasible movement is chosen.

As already noted, in each neighbourhood an ordered search tests all possible combinations of pairs of routes (R_p, R_q); a total number of $v(v-1)/2$ different pairs of routes (R_p, R_q) are potentially tested, where v is the number of routes. The customers in each route are searched sequentially.

In addition to the three neighbourhoods defined above, a fourth neighbourhood was defined by sequentially searching these three neighbourhoods. This combined neighbourhood was also explored both on a *first admissible* and on *best admissible movement* basis.

The *best admissible movement* always produced better results. For this reason it was always used, in all neighbourhoods tested by our TS algorithm.

Only feasible movements are considered. After completing a neighbourhood search, a 2-opt movement is executed for each route, in an attempt to further reduce the total length of the routes.

4.4. Short-term memory

A recency-based memory structure was implemented to avoid cycling. Every movement is characterized by two sets of edges, which define its attributes: (i) edges to be inserted into the solution; (ii) edges to be removed from the solution. The short-term memory maintains a list of the edges that were used (inserted or removed) in the recent past. The same tabu list is used to save both inserted and removed edges, since it is always possible to identify if an edge was inserted or removed by observing the current solution.

A movement is considered tabu when all the edges involved are tabu in the current iteration. This is not so restrictive, in the sense that movements can be made even when some (but not all) of the associated edges are classified as tabu.

Tabu tenure: Different tabu tenure values were assigned to edges inserted or removed. Inserted edges have lower tenures than removed edges. We initially tested variable tabu tenures, chosen from uniform distributions defined in the intervals (10,20) for inserted edges and (30,50) for removed edges. We discovered however that better results were obtained with the use of fixed tabu tenures. Consequently, in its current implementation a movement executed in iteration t is forbidden until iteration $t+q$, where q is a tabu tenure value chosen proportionally to the number of nodes. Note that different values of q are used for inserted and removed edges. The aspiration criterion we use overrides the tabu status of a movement if this leads to a better solution than the best found so far.

4.5. Long-term memory

A frequency-based memory was implemented to keep a history of the most frequently used edges (residence frequency). This information was used to intensify/diversify the tabu search. The residence frequency of an inserted or removed arc is "boosted" by an *incentive*, so that, in the intensification phase of the search, it becomes attractive to insert arcs with high frequencies and remove arcs with low frequencies, and, in the diversification phase, it becomes attractive to remove arcs with high frequencies and insert arcs with low frequencies. The value of the *incentive* was defined as the average of all arcs in the *distance matrix* of the network.

The intensification and diversification phases of the tabu search are controlled by what we call the *economy of a movement*. Expressions for this *economy* are given below for both the intensification and diversification phases.

Intensification: Consider the following example of the relocation movement shown in Fig. 3, in which arcs (p_{i-1}, p_i) and (p_i, p_{i+1}) are removed from route R_p and arc (q_{j-1}, q_j) is removed from route R_q, being replaced, respectively, by arc (p_{i-1}, p_{i+1}) in route R_p and by arcs (q_{j-1}, p_i) and (p_i, q_j) in route R_q. In this case the *economy of the movement* (eco_mov) is defined as

$$\begin{aligned}\text{eco\_mov}_i = & l(p_{i-1}, p_i) + [1 - f(p_{i-1}, p_i)]^* \text{ incentive} \\ & + l(p_i, p_{i+1}) + [1 - f(p_i, p_{i+1})]^* \text{ incentive} \\ & + l(q_{j-1}, q_j) + [1 - f(q_{j-1}, q_j)]^* \text{ incentive} \\ & - l(p_{i-1}, p_{i+1}) + f(p_{i-1}, p_{i+1})^* \text{ incentive} \\ & - l(q_{j-1}, p_i) + f(q_{j-1}, p_i)^* \text{ incentive} \\ & - l(p_i, q_j) + f(p_i, q_j)^* \text{ incentive},\end{aligned}$$

where $l(a1, a2)$ is the length of arc (a1,a2) and $f(a1, a2)$ is the *residence frequency* of arc (a1,a2). The largest value of eco_mov_i defines the intensification movement to be made.

Diversification: Consider now the same example of Fig. 3 in the case of a diversification movement. The corresponding formula for eco_mov_d is given by

$$\begin{aligned}\text{eco\_mov}_d = & l(p_{i-1}, p_i) + f(p_{i-1}, p_i)^* \text{incentive} \\ & + l(p_i, p_{i+1}) + f(p_i, p_{i+1})^* \text{incentive} \\ & + l(q_{j-1}, q_j) + f(q_{j-1}, q_j)^* \text{incentive} \\ & - l(p_{i-1}, p_{i+1}) + [1 - f(p_{i-1}, p_{i+1})]^* \text{incentive} \\ & - l(q_{j-1}, p_i) + [1 - f(q_{j-1}, p_i)]^* \text{incentive} \\ & - l(p_i, q_j) + [1 - f(p_i, q_j)]^* \text{incentive}.\end{aligned}$$

4.6. General structure of our tabu search algorithm for VRP_SPD

Step 0 (*Initialization*): Generate an initial solution by using one of the following heuristic procedures: IGR1, IGR2, SGR1, SGR2.

Step 1 (*Initial phase*): Execute the tabu search algorithm beginning from the current initial solution. The algorithm is executed exactly ini_{iter} iterations.

Step 2 (*Intensification phase*): Execute the intensification procedure of the tabu search algorithm beginning from the best solution found in Step 1. Original tabu tenure values are divided by two during intensification. The algorithm is executed exactly i_{iter} iterations.

Step 3 (*Diversification phase*): Execute the diversification procedure of the tabu search algorithm beginning from the solution obtained in Step 2. Tabu tenure values used in the previous phase are maintained during the diversification phase. The algorithm is executed exactly d_{iter} iterations.

Step 4 (*Standard phase*): Re-execute the original tabu search algorithm beginning from the solution obtained in Step 3. Original tabu tenure values for removed and inserted arcs are re-established. The algorithm is executed exactly $inter_{\text{iter}}$ iterations.

Step 5 (*Interactive phase*): Repeat Steps 2–4 until one of the stopping rules is reached.

Step 6 (*Re-initialization*): Generate a new initial solution by utilizing one of the heuristics (IGR1, IGR2, SGR1, SGR2) not used so far. Repeat Steps 1–5 until all the four heuristic procedures are utilized to produce an initial solution.

Stopping rules:

- no feasible movement exists;
- maximum number of iterations for the corresponding phase was reached (Steps 1–4). Note that these values are the same for each re-initialization of the search (Steps 2–4).

Parameter values: The following parameter values were used:

$$ini_{\text{iter}} = 5000;$$
$$i_{\text{iter}} = 500;$$
$$d_{\text{iter}} = 200;$$
$$inter_{\text{iter}} = 2000.$$

5. Computational results

The tabu search algorithm we developed was implemented in Pascal for the Delphi 5.0 environment and run on an Athlon XP 2.0 GHz PC with 256 MB of RAM. For testing the algorithm we used data available in the literature for VRP_SPD, in order to be able to compare the quality of our solutions with that of other algorithms developed for the same problem. These are the algorithms of Dethloff [3] and Salhi and Nagy [4,7], which, to the best of our knowledge, are the only ones specifically dedicated to VRP_SPD as defined in the present paper. These authors were kind enough to provide us with their data and corresponding computational results.

On the other hand, given the size of the Solomon and Extended Solomon test problems (see Solomon [40], Gehring and Homberger [41]), we also tested our algorithm with a subset of these problems, although in this case no direct comparisons were possible, since the algorithms of Dethloff [3] and Salhi and Nagy [4,7] do not use these data. The data of [40,41] had to be adapted for VRP_SPD, given that only one demand is defined for each client in these problems The adaptation of these data consisted of randomly generating discrete pick-up demands, in the interval used by the authors of [40,41] to generate delivery demand.

Table 1
Comparison between different neighbourhoods: mean improvement over initial solution

Data sets	% Improvement			
	Relocation	Interch.	Crossover	Combined
Dethloff	6.18	3.88	**8.01**	7.65
Salhi & Nagy without dist. constraints	9.27	5.26	10.39	**10.61**
Salhi & Nagy with dist. constraints	10.42	5.74	10.75	**11.41**
Solomon & Extended Solomon	7.99	3.37	8.26	**8.67**
Min	2.54	6.42	4.59	**7.44**

Table 2
Comparison between different neighbourhoods: percentage of best solutions obtained

Data sets	% Best solutions			
	Relocation	Interch.	Crossover	Combined
Dethloff	8.75	0.63	**56.88**	53.13
Salhi & Nagy without dist. constraints	19.64	0.00	35.71	**53.57**
Salhi & Nagy with dist. constraints	25.00	0.00	32.14	**46.43**
Solomon & Extended Solomon	18.06	0.00	22.22	**62.50**
Min	0.00	25.00	0.00	**100.00**

In total, we used 87 test problems to evaluate our algorithm: 40 from Dethloff [3], 28 from Salhi and Nagy [4], 18 from Solomon [40] and Gehring and Homberger [41] and one from Min [1]. For each of these problems we run the Tabu Search Algorithm defined in Section 4.6 for each of the four neighbourhoods defined in Sections 4.2 and 4.3. We now compare these results in order to determine the *best neighbourhood* created by our algorithm. For this purpose we adopted two criteria, the percentage of improvement over the initial solution and the percentage of best solutions obtained. We use the *best neighbourhood*, as defined by the two above criteria, to compare our algorithm with the algorithms of Dethloff [3] and Salhi and Nagy [4,7].

In Table 1, we show summarized data on improvements we obtained over the initial solution of the tabu search, for each of the 4 movements and each of the data sets we used (the % improvement shown in this table is given by (Initial solution−heuristic solution)/Initial solution * 100%). In general we observed a slight superiority of the combined movement over the three single movements, particularly for the larger instances.

In Table 2, we show percentages of best solutions obtained by each of the 4 movements and each of the data sets used. This table confirms the superiority of the combined movement. Given these results, we only show solutions produced by the combined movement in Tables 3–12.

Table 3 compares the best results obtained by Dethloff [3] (with his RCRS heuristic) with the results obtained by the tabu search we developed (combined movement, as already explained) for each one of the 40 Dethloff problems (no maximum distance constraints are used by Dethloff). In his paper, Dethloff does not provide data on the number of vehicles used or on computing times. The last column of this table

Table 3
Comparison with the Dethloff algorithm (RCRS) [3]

Problem	No. of clients	Dethloff [3]			TS—combined movement			% Improv. over RCRS
		No. of vehicles	RCRS sol. value	RCRS C time	No. of vehicles	TS sol. value	TS C time*	
SCA3-0	50	—	689.00	—	4	640.55	3.37	7.03
SCA3-1	50	—	765.60	—	4	697.84	3.25	8.85
SCA3-2	50	—	742.80	—	4	659.34	3.52	11.24
SCA3-3	50	—	737.20	—	4	680.04	3.31	7.75
SCA3-4	50	—	747.10	—	4	690.50	3.43	7.58
SCA3-5	50	—	784.40	—	4	659.90	3.67	15.87
SCA3-6	50	—	720.40	—	4	653.81	3.35	9.24
SCA3-7	50	—	707.90	—	4	659.17	3.33	6.88
SCA3-8	50	—	807.20	—	4	719.47	3.40	10.87
SCA3-9	50	—	764.10	—	4	681.00	3.41	10.88
SCA8-0	50	—	1132.90	—	9	981.47	4.14	13.37
SCA8-1	50	—	1150.90	—	9	1077.44	4.27	6.38
SCA8-2	50	—	1100.80	—	10	1050.98	4.20	4.53
SCA8-3	50	—	1115.60	—	9	983.34	4.17	11.86
SCA8-4	50	—	1235.40	—	9	1073.46	4.13	13.11
SCA8-5	50	—	1231.60	—	9	1047.24	4.02	14.97
SCA8-6	50	—	1062.50	—	9	995.59	3.85	6.30
SCA8-7	50	—	1217.40	—	10	1068.56	4.22	12.23
SCA8-8	50	—	1231.60	—	9	1080.58	3.85	12.26
SCA8-9	50	—	1185.60	—	9	1084.80	4.20	8.50
CON3-0	50	—	672.40	—	4	631.39	3.64	6.10
CON3-1	50	—	570.60	—	4	554.47	3.31	2.83
CON3-2	50	—	534.80	—	4	522.86	3.45	2.23
CON3-3	50	—	656.90	—	4	591.19	3.28	10.00
CON3-4	50	—	640.20	—	4	591.12	3.47	7.67
CON3-5	50	—	604.70	—	4	563.70	3.38	6.78
CON3-6	50	—	521.30	—	4	506.19	3.32	2.90
CON3-7	50	—	602.80	—	4	577.68	3.51	4.17
CON3-8	50	—	556.20	—	4	523.05	3.66	5.96
CON3-9	50	—	612.80	—	4	580.05	3.36	5.34
CON8-0	50	—	967.30	—	9	860.48	4.19	11.04
CON8-1	50	—	828.70	—	9	740.85	3.89	10.60
CON8-2	50	—	770.20	—	9	723.32	3.76	6.09
CON8-3	50	—	906.70	—	10	811.23	4.12	10.53
CON8-4	50	—	876.80	—	9	772.25	3.75	11.92
CON8-5	50	—	866.90	—	9	756.91	3.99	12.69
CON8-6	50	—	749.10	—	9	678.92	4.04	9.37
CON8-7	50	—	929.80	—	9	814.50	4.00	12.40
CON8-8	50	—	833.10	—	9	775.59	3.74	6.90
CON8-9	50	—	877.30	—	9	809.00	4.13	7.79
					Minimum			2.23
					Mean			8.82
					Maximum			15.87

*CPU seconds in an Athlon 2.0 GHz PC.

Table 4
Comparison between different algorithms used for the Min problem [1]

Problem	No. of clients	Min			Dethloff			TS—combined movement			% Improv. over Min	% Improv. over RCRS
		No. of vehicles	Min sol. value	Min C time	No. of vehicles	RCRS sol. value	RCRS C time	No. of vehicles	TS sol. value	TS C time*		
MIN22	22	2	94	—	2	91	—	2	88	1.2	6.38	3.30

*CPU seconds in an Athlon 2.0 GHz PC.

Table 5
Comparison between different algorithms for the Salhi and Nagy Problems (without maximum distance constraints) [4]

Problem	No. of clients	Salhi & Nagy			Dethloff			TS—combined movement			% Improv. over LC	% Improv. over RCRS
		No. of vehicles	LC sol. value	LC C time*	No. of vehicles	RCRS sol. value	RCRS C time	No. of vehicles	TS sol. value	TS C time**		
CMT1X	50	6	601	3	3	501	—	3	472	3.73	21.46	5.79
CMT1Y	50	5	603	3	3	501	—	3	470	4.37	22.06	6.19
CMT2X	75	11	903	1.7	7	782	—	7	695	6.91	23.03	11.13
CMT2Y	75	12	924	1.3	7	782	—	7	700	7.61	24.24	10.49
CMT3X	100	10	923	2.3	5	847	—	5	721	11.04	21.89	14.88
CMT3Y	100	10	923	2.3	5	847	—	5	719	12.01	22.10	15.11
CMT12X	100	10	831	4.9	6	804	—	6	675	12.23	18.77	16.04
CMT12Y	100	11	873	4.8	5	825	—	6	689	12.80	21.08	16.48
CMT11X	120	11	1500	3.4	4	959	—	4	900	18.17	40.00	6.15
CMT11Y	120	11	1500	3.6	4	1070	—	5	910	18.04	39.33	14.95
CMT4X	150	15	1178	4.3	7	1050	—	7	880	24.60	25.30	16.19
CMT4Y	150	15	1178	4.3	7	1050	—	7	878	29.07	25.47	16.38
CMT5X	199	19	1509	12.8	11	1348	—	11	1098	51.50	27.24	18.55
CMT5Y	199	19	1477	12.9	11	1348	—	10	1083	56.21	26.68	19.66
								Minimum			18.77	5.79
								Mean			25.62	13.43
								Maximum			40.00	19.66

*CPU seconds in a VAX 4000-500.
**CPU seconds in an Athlon 2.0 GHz PC.

shows the % improvement we obtained over Dethloff's RCRS heuristic (%improvement = ((|(TS value) − (RCRS value)|/RCRS value) ∗ 100)); these results are summarized at the bottom of the table. The mean improvement obtained for the 40 problems was 8.82%.

Table 4 compares results obtained by three algorithms developed for VRP_SPD (the algorithms of Min [1], RCRS of Dethloff [3] and our TS) for the Min problem [1]. Our TS algorithm obtained the optimal solution of this problem (see Section 5.1).

Tables 5 and 6 compare results obtained by three algorithms developed for VRP_SPD using Salhi and Nagy data: The algorithms of Salhi & Nagy [4], the RCRS heuristic of Dethloff [3] and our tabu search.

Table 6
Comparison between different algorithms for the Salhi and Nagy Problems (with maximum distance constraints) [4]

Problem	No. of clients	Salhi & Nagy			Dethloff			TS—combined movement			%Improv. over LC	% Improv. over RCRS
		No. of vehicles	LC sol. value	LC C time*	No. of vehicles	RCRS sol. value	RCRS C time	No. of vehicles	TS sol. value	TS C time**		
CMT6X	50	6	601	3.0	6	584	—	3	476	1.29	20.80	18.49
CMT6Y	50	6	609	1.7	6	584	—	3	474	1.30	22.17	18.84
CMT7X	75	—	—	—	11	961	—	7	695	4.01	—	27.68
CMT7Y	75	—	—	—	11	961	—	6	700	3.21	—	27.16
CMT8X	100	10	923	2.3	9	928	—	5	720	6.05	21.99	22.41
CMT8Y	100	10	927	2.2	9	936	—	5	721	6.31	22.22	22.97
CMT14X	100	13	954	5.0	10	871	—	6	675	6.21	29.25	22.50
CMT14Y	100	12	920	5.3	10	871	—	6	689	6.42	25.11	20.90
CMT13X	120	13	1613	3.0	11	1576	—	5	918	9.41	43.09	41.75
CMT13Y	120	12	1589	2.8	11	1576	—	5	910	9.50	42.73	42.26
CMT9X	150	15	1215	4.6	15	1299	—	7	885	12.52	27.16	31.87
CMT9Y	150	15	1215	4.6	15	1299	—	8	900	14.66	25.93	30.72
CMT10X	199	21	1573	17.0	19	1571	—	11	1100	25.14	30.07	29.98
CMT10Y	199	20	1527	17.2	19	1571	—	11	1083	28.42	29.08	31.06
								Minimum			20.80	18.49
								Mean			28.30	27.76
								Maximum			43.09	42.26

*CPU seconds in a VAX 4000-500.
**CPU seconds in an Athlon 2.0 GHz PC.

Table 5 corresponds to problems without maximum distance constraints, whereas maximum distance constraints are present in the problems of Table 6. Recently, Salhi and Nagy [7] improved their initial algorithm published in [4], but in their latter paper results are not presented by individual problem (only average results for groups of problems are given). The results shown in Tables 5 and 6 are those of their earlier paper.

The general conclusions that can be drawn from Tables 5 and 6 is that our TS algorithm represents an improvement over the two other algorithms for the Salhi and Nagy problems, as it can be easily seen from the summarized results presented at the bottom of both tables. These conclusions do not change in view of the more recent results of Salhi and Nagy (see [7]), judging from the average results presented in that paper.

The improvement we obtained is particularly important for the problems with maximum distance constraints (see Table 6). In this case, our TS algorithm not only improved the mean route costs obtained by the two other heuristics by approximately 27%, but also consistently used considerably less vehicles than these two other algorithms.

Finally, the results of the tabu search for selected Solomon and Extended Solomon problems are shown in Table 7. Although no comparisons with other algorithms are possible for these problems, it is interesting to observe how computing times increase with problem size. Even the largest 400-client problems were run in a time below 340 s of CPU.

Table 7
Tabu Search results for selected Solomon and extended Solomon problems [40,41]

Problem	TS—combined movement			
	No. of clients	No. of vehicles	TS sol. value	TS C time*
r101	100	12	1042.62	13.20
r201	100	3	671.03	12.02
c101	100	17	1259.79	12.07
c201	100	5	666.01	12.40
rc101	100	11	1094.15	12.30
rc201	100	3	674.46	12.07
r1_2_1	200	23	3447.20	55.56
r2_2_1	200	5	1690.67	50.95
c1_2_1	200	29	3792.62	52.21
c2_2_1	200	9	1767.58	65.79
rc1_2_1	200	24	3427.19	58.39
rc2_2_1	200	5	1645.94	52.93
r1_4_1	400	54	10027.81	330.42
r2_4_1	400	10	3695.26	324.44
c1_4_1	400	65	11676.27	287.12
c2_4_1	400	15	3732.00	330.20
rc1_4_1	400	52	9883.31	286.66
rc2_4_1	400	11	3603.53	328.16

*CPU seconds in an Athlon 2.0 GHz PC.

5.1. Comparison with lower bounds

The final appraisal of any heuristic method rests on how close solutions produced by the proposed methodology come to the optimal solutions of the problems under consideration. It is not always possible, of course, to obtain optimal solutions to hard combinatorial problems and for this reason comparisons are often made with corresponding lower bounds (for minimization problems). In the case of VRP_SPD we initially obtained lower bounds from Lagrangean relaxations of the compact mathematical formulation described in Section 3. The results, however, were disappointing: the average gap between heuristic solutions (upper bounds) and corresponding lower bounds, for problems between 30 and 80 vertices, was between 22% and 29%.

We decided to obtain lower bounds by an alternative methodology, running the models of Section 3 using version 9.0 of CPLEX. Since these are hard combinatorial problems, it is doubtful whether CPLEX can find corresponding optimal solutions in reasonable computing times. We adopted therefore the strategy of using CPLEX to run each of the test problems for a pre-defined amount of time (2 h) in a Pentium IV 2.4 GHz PC with 512 MB of RAM.

CPLEX parameters were set such that the emphasis of the search was on finding the best possible lower bound within the pre-defined time. A solution obtained through our TS algorithm was always provided as an initial solution. After the pre-defined time, processing of each problem was interrupted and the value of the best lower bound obtained by CPLEX written to file, before the next test problem was executed. The corresponding computational results are shown in Tables 8–12.

Table 8
Lower bounds for the Dethloff Problems [3]

Problem	No. of clients	TS sol. value	CPLEX LB value	% Gap
SCA3-0	50	640.55	583.77	8.86
SCA3-1	50	697.84	655.63	6.05
SCA3-2	50	659.34	627.12	4.89
SCA3-3	50	680.04	633.56	6.84
SCA3-4	50	690.50	642.89	6.89
SCA3-5	50	659.90	603.06	8.61
SCA3-6	50	653.81	607.53	7.08
SCA3-7	50	659.17	616.40	6.49
SCA3-8	50	719.47	668.04	7.15
SCA3-9	50	681.00	619.03	9.10
SCA8-0	50	981.47	877.55	10.59
SCA8-1	50	1077.44	954.29	11.43
SCA8-2	50	1050.98	950.74	9.54
SCA8-3	50	983.34	905.29	7.94
SCA8-4	50	1073.46	972.62	9.39
SCA8-5	50	1047.24	940.60	10.18
SCA8-6	50	995.59	885.34	11.07
SCA8-7	50	1068.56	955.86	10.55
SCA8-8	50	1080.58	986.52	8.70
SCA8-9	50	1084.80	978.90	9.76
CON3-0	50	631.39	592.38	6.18
CON3-1	50	554.47	532.55	3.95
CON3-2	50	522.86	491.04	6.09
CON3-3	50	591.19	557.99	5.62
CON3-4	50	591.12	558.26	5.56
CON3-5	50	563.70	531.33	5.74
CON3-6	50	506.19	475.33	6.10
CON3-7	50	577.68	550.73	4.67
CON3-8	50	523.05	492.69	5.80
CON3-9	50	580.05	547.31	5.65
CON8-0	50	860.48	795.45	7.56
CON8-1	50	740.85	693.22	6.43
CON8-2	50	723.32	650.81	10.02
CON8-3	50	811.23	754.41	7.00
CON8-4	50	772.25	729.09	5.59
CON8-5	50	756.91	709.76	6.23
CON8-6	50	678.92	631.41	7.00
CON8-7	50	814.50	762.03	6.44
CON8-8	50	775.59	705.08	9.09
CON8-9	50	809.00	729.10	9.88
		Minimum		3.95
		Mean		7.54
		Maximum		11.43

Table 9
Lower bounds for the Min problem [1]

Problem	No. of clients	TS sol. values	CPLEX LB value	% Gap
MIN22	22	88	88	0.00

Table 10
Lower bounds for the Salhi and Nagy Problems [4] (without maximum distance constraints)

Problem	No. of clients	TS sol. values	CPLEX LB value	% Gap
CMT1X	50	472	454.68	3.67
CMT1Y	50	470	455.52	3.08
CMT2X	75	695	617.01	11.22
CMT2Y	75	700	617.64	11.77
CMT3X	100	721	646.73	10.30
CMT3Y	100	719	648.04	9.87
CMT12X	100	675	568.79	15.73
CMT12Y	100	689	573.53	16.76
CMT11X	120	900	663.38	26.29
CMT11Y	120	910	662.84	27.16
CMT4X	150	880	714.18	18.84
CMT4Y	150	878	715.67	18.49
CMT5X	199	1098	858.14	21.85
CMT5Y	199	1083	856.59	20.91
		Minimum		3.08
		Mean		15.42
		Maximum		27.16

Results for the Dethloff problems [3] are shown in Table 8. The last column of this table measures the %gap between the TS solution (combined movement) and the lower bound obtained by CPLEX (%gap = ((TS solution − lower bound)/TS solution)*100). The results are summarized at the bottom of the table. For these problems the mean and the maximum percentage gaps were, respectively, 7.54% and 11.43%.

It is worth emphasizing that these gaps are the *maximum errors* that can be made when optimal solutions are estimated using our TS algorithm. The *real errors* embedded in these heuristic solutions (which could only be obtained if CPLEX could find the corresponding optimal solutions) are expected to be much lower than the gaps shown in the tables.

In Table 9 we show results corresponding to Min problem [1]. It is important to note that CPLEX was able to find an optimal solution for this problem (reported for the first time). As already noted, our TS algorithm also found this optimal solution.

In Tables 10 and 11, we show results corresponding to the Salhi and Nagy problems [4] (without and with maximum distance constraints, respectively). In this case, due to the larger size of these problems, the mean percentage gaps are bigger than those obtained for the Dethloff problems: 15.42% and 11.94%, respectively, for Tables 10 and 11.

Table 11
Lower bounds for the Salhi and Nagy Problems [4] (with maximum distance constraints)

Problem	No. of clients	TS sol. value	CPLEX LB value	% Gap
CMT6X	50	476	436.63	8.27
CMT6Y	50	474	437.11	7.78
CMT7X	75	695	607.97	12.52
CMT7Y	75	700	606.38	13.37
CMT8X	100	720	649.25	9.83
CMT8Y	100	721	655.53	9.08
CMT14X	100	675	558.86	17.21
CMT14Y	100	689	568.48	17.49
CMT13X	120	918	—	—
CMT13Y	120	910	—	—
CMT9X	150	885	—	—
CMT9Y	150	900	—	—
CMT10X	199	1100	—	—
CMT10Y	199	1083	—	—
		Minimum		7.78
		Mean		11.94
		Maximum		17.49

Table 12
Lower bounds for selected Solomon and extended Solomon problems [40,41]

Problem	No. of clients	TS sol. value	CPLEX LB value	% Gap
r101	100	1042.62	934.97	10.33
r201	100	671.03	643.65	4.08
c101	100	1259.79	1066.19	15.37
c201	100	666.01	596.85	10.38
rc101	100	1094.15	937.41	14.33
rc201	100	674.46	602.70	10.64
r1_2_1	200	3447.20	2951.12	14.39
r2_2_1	200	1690.67	1501.82	11.17
c1_2_1	200	3792.62	3299.07	13.01
c2_2_1	200	1767.58	1542.96	12.71
rc1_2_1	200	3427.19	2939.98	14.22
rc2_2_1	200	1645.94	1396.95	15.13
r1_4_1	400	10027.81	8256.91	17.66
r2_4_1	400	3695.26	3068.77	16.95
c1_4_1	400	11676.27	10245.85	12.25
c2_4_1	400	3732.00	3047.91	18.33
rc1_4_1	400	9883.31	8387.71	15.13
rc2_4_1	400	3603.53	2941.98	18.36
		Minimum		4.08
		Mean		13.58
		Maximum		18.36

Finally, in Table 12 we show results corresponding to selected Solomon and extended Solomon problems [40,41]. In this case, the mean and the maximum percentage gaps were 13.58% and 18.36%, respectively.

6. Conclusions

In this paper we developed a tabu search algorithm to solve VRP_SPD. This algorithm uses three types of movements to obtain inter-route adjacent solutions and a *2-opt* procedure to obtain alternative intra-route solutions. Four types of neighbourhoods were implemented, and two different search strategies were tested for selecting the next movement. Intensification and diversification of the search were achieved through frequency penalization. Computational results are reported for a set of 87 test problems with between 50 and 400 clients.

The combined movement produced in general the best results: except for the Dethloff problems it produced both the best "% improvement" over the initial TS solution and the best "% of best solutions" found by each of the four movements. For the Dethloff problems the crossover movement produced slightly better results; in this case the superiority of the crossover movement may be attributed to the relatively small problem size (50 nodes) and to the use of real values for both distances between clients and clients'demands. The combined movement, on the other hand, produced better results for the larger problems, in which generally integer values were used for distances and demands. These conclusions would however have to be confirmed by further testing our procedure.

It was our experience that neither intensification nor diversification of the TS improved significantly the final solution provided by the algorithm, for all data sets we used. There were instances in which some improvement was provided by theses strategies, but in other instances no improvement was achieved by their use.

Our proposed TS procedure represents an improvement over former heuristics developed for the same problem. It produced better results for all data sets tested by both Dethloff [3] and Salhi and Nagy [4]. The mean % improvement we obtained for these three data sets were, respectively, 8.82% (Dethloff problems), 13.43% (Salhi and Nagy problems, without distance constraints) and 27.76% (Salhi and Nagy problems, with distance constraints). The computing times were generally low, with a maximum CPU time below 340 s for the larger problems (400-node Solomon and Extended Solomon problems).

Acknowledgements

We are indebted to Dr. Dethloff and Dr. Salhi for having kindly provided us with their data and corresponding computational results. This research was sponsored by the Brazilian National Research Council (CNPq), Grant No. 472295/2003-9.

References

[1] Min H. The multiple vehicle routing problem with simultaneous delivery and pickup points. Transportation Research A 1989;23:377–86.

[2] Galvão RD, Guimarães J. The control of helicopter operations in the Brazilian oil industry: issues in the design and implementation of a computerized system. European Journal of Operational Research 1990;49:266–70.

[3] Dethloff J. Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. OR Spektrum 2001;23:79–96.
[4] Salhi S, Nagy G. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. Journal of the Operational Research Society 1999;50:1034–42.
[5] Golden BL, Baker E, Alfaro J, Schaffer J. The vehicle routing problem with backhauling: two approaches. Proceedings of the Twenty-First Annual Meeting of the S. E. TIMS, Myrtle Beach, SC, USA, 1985.
[6] Casco DO, Golden BL, Wasil EA. Vehicle routing with backhauls. In: Golden BL, Assad AA, editors. Vehicle routing: methods and studies. Amsterdam: North-Holland; 1988. p. 127–47.
[7] Salhi S, Nagy G. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. European Journal of Operational Research 2004. Available online at www.sciencedirect.com.
[8] Tang FA, Galvão RD. Vehicle routing problems with simultaneous pick-up and delivery service. Journal of the Operational Research Society of India (OPSEARCH) 2002;39:19–33.
[9] Beasley JE. Route first-cluster second methods for vehicle routing. Omega 1983;11:403–8.
[10] Gillett BE, Miller LR. A heuristic algorithm for the vehicle dispatch problem. Operations Research 1974;22:340–9.
[11] Angelelli E, Mansini R. A branch-and-price algorithm for a simultaneous pick-up and delivery problem. Working Paper, Article presented at the EURO/INFORMS Meeting, 2003.
[12] Tang FA, Ferreira Filho VJM, Galvão RD. Determinação de rotas para empresas de entrega expressa ("Design of routes for express delivery companies"). Journal of the Brazilian OR Society (Pesquisa Operacional) 1997;17:107–35.
[13] Savelsbergh MWP, Sol M. The general pickup and delivery problem. Transportation Science 1995;29:17–29.
[14] Psaraftis H. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science 1980;14:130–54.
[15] Psaraftis H. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. Transportation Science 1983;17:351–60.
[16] Desrosiers J, Dumas Y, Soumis F. A dynamic programing solution of a large scale single vehicle dial-a-ride problem with time windows. American Journal of Mathematics and Management Science 1986;6:301–26.
[17] Psaraftis H. K-interchange procedures for local search in a precedence-constrained routing problem. European Journal of Operational Research 1983;13:391–402.
[18] Sexton T, Bodin L. Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. Transportation Science 1985;19:378–410.
[19] Sexton T, Bodin L. Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing. Transportation Science 1985;19:411–35.
[20] Sexton T, Choi Y. Pickup and delivery of partial loads with time windows. American Journal of Mathematics and Management Science 1986;6:369–98.
[21] Van der Bruggen LJJ, Lenstra JK, Schuur PC. Variable-depth search for the single-vehicle pickup and delivery problem with time windows. Transportation Science 1993;27:298–311.
[22] Madsen O, Ravn H, Rygaard JR. REBUS, A system for dynamic vehicle routing for the Copenhagen Fire Fighting Company. Research Report 2/1993, IMSOR, Lyngby, Denmark, 1993.
[23] Roy S, Rousseau JM, Lapalme G, Ferland JA. Routing and scheduling for the transportation of disabled persons—the algorithm. Report TP 5596E, Transport Development Center, Montréal, Canada, 1984.
[24] Roy S, Rousseau JM, Lapalme G, Ferland JA. Routing and scheduling for the transportation of disabled persons—the tests. Report TP 5596E, Transport Development Center, Montréal, Canada, 1984.
[25] Jaw JJ, Odoni AR, Psaraftis HN, Wilson NHM. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research B 1986;20:243–57.
[26] Desrosiers J, Dumas Y, Soumis F, Taillefer S, Villeneuve D. An algorithm for mini-clustering in handicapped transport. Cahiers du GERARD Report G-91-02, École des Hautes Études Commerciales, Montréal, Canada, 1991.
[27] Ioachim I, Desrosiers J, Dumas Y, Solomon MM, Villeneuve D. A request clustering algorithm for door-to-door handicapped transportation. Transportation Science 1995;29:63–78.
[28] Dumas Y, Desrosiers J, Soumis F. The pickup and delivery problem with time windows. European Journal of Operational Research 1991;54:7–22.
[29] Gendreau M, Hertz A, Laporte G. The traveling salesman problem with backhauls. Computers and Operations Research 1996;23:501–8.

[30] Deif I, Bodin LD. Extension of the Clarke and Wright algorithm for solving the vehicle routing with backhauling. Proceedings of the Babson Conference on Software Uses in Transportation and Logistics Management, Babson Park, MA, 1984.
[31] Toth P, Vigo D. An exact algorithm for the vehicle routing problem with backhauls. Transportation Science 1997;31: 372–85.
[32] Mingozzi A, Giorgi S. An exact method for the vehicle routing problem with backhauls. Transportation Science 1999;33: 315–29.
[33] Gélinas S, Desrochers M, Desrosiers J, Solomon MM. A new branching strategy for time constrained routing problems with application to backhauling. Annals of Operations Research 1995;61:91–110.
[34] Duhamel C, Potvin JY, Rousseau JM. A tabu search heuristic for the vehicle routing problem with backhauls and time windows. Transportation Science 1997;31:49–59.
[35] Mosheiov G. The traveling salesman problem with pick-up and delivery. European Journal of Operational Research 1994;79:299–310.
[36] Mosheiov G. Vehicle routing with pick-up and delivery: tour partitioning heuristics. Computers and Industrial Engineering 1998;34:669–84.
[37] Gavish B, Graves S. The travelling salesman problem and related problems. Working Paper 7905, Graduate School of Management, University of Rochester, NY, USA, 1979.
[38] Anily S, Mosheiov G. The traveling salesman problem with delivery and backhauls. Operations Research Letters 1994;16: 11–8.
[39] Gendreau M, Laporte G, Vigo D. Heuristics for the traveling salesman problem with pickup and delivery. Computers and Operations Research 1999;26:699–714.
[40] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 1987;35:254–65.
[41] Gehring H, Homberger J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Miettinen K, Mäkelä M, Toivanen J, editors. Proceedings of EUROGEN99, vol. A2(S), Springer, Berlin, 1999, pp. 57–64.