A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows and Split Deliveries

Sin C. Ho* and Dag Haugland†

Department of Informatics
University of Bergen
N-5020 Bergen, Norway

4th September 2002

Abstract

In this paper, we consider the Vehicle Routing Problem with Time Windows and Split Deliveries (VRPTWSD) and present a solution method based on Tabu Search and four different neighborhood structures. This problem is an extension of the Vehicle Routing Problem, where the start of service at each customer must be within a time window, and a customer may be serviced by more than one vehicle. The heuristic gives promising results when applied to realistic instances of the problem. It is also adapted to the problem where splitting is not an option, and experimental results have shown that the heuristic has improved 5 of the 56 best published solutions to the Solomon benchmarks, while matching or improving the best solutions in 10 problems.

Keywords: Vehicle Routing, Time Windows, Split Deliveries, Tabu Search

1 Introduction

Vehicle Routing Problems (VRP) are concerned with the distribution of goods, people or information between depots and customers. Vehicle routing problems arise in many real-life applications within transportation and logistics, such as school bus routing, postal deliveries, transportation of handicapped persons and food distribution. This paper considers the Vehicle Routing Problem with Time Windows and Split Deliveries (VRPTWSD). Given a fleet of homogeneous vehicles stationed at a central depot and a set of customers requiring their demands to be fulfilled, the problem consists of finding vehicle routes starting and ending at the depot such that every customer is visited. The routes must also meet the time windows defined by the customers, which specify when the start of service can occur. Sometimes it is not realistic that a customer's demand must be delivered by a single vehicle. By allowing deliveries to be split, a customer may be serviced by more than one vehicle. The objective of the VRPTWSD consists of minimizing the total distance traveled (equivalent to cost minimization).

Since the VRPTWSD is NP-hard (see Lenstra and Rinnooy Kan 1981, Dror and Trudeau 1990), instances of realistic size are difficult to solve to optimality. As a way out, metaheuristics are often used to find good solutions to various routing problems in reasonable time.

*Email: sin@ii.uib.no

†Email: dag.haugland@ii.uib.no

The past years, quite good results have been achieved for the Vehicle Routing Problem with Time Windows (VRPTW), in both the classes of exact methods and metaheuristics. Surveys can be found in Toth and Vigo (2002, Chapter 7) and Golden and Assad (1988). Bräysy and Gendreau (2001) give an excellent overview over metaheuristics for the VRPTW. All of these considered traditionally vehicle routing for which each customer is visited exactly once.

The Vehicle Routing Problem with Split Deliveries (VRPSD) is introduced by Dror and Trudeau in 1989. They showed how split deliveries could result in savings, both in the total distance traveled and the number of vehicles utilized. The VRPSD is a relaxation of the classical Vehicle Routing Problem, but it remains NP-hard (see Dror and Trudeau 1990). Dror et al. (1994) described an integer programming formulation of the problem and developed several classes of valid constraints. They also developed an exact constraint relaxation branch and bound algorithm for the VRPSD. Frizzell and Giffin (1992, 1995) studied the problem with grid network distances, whereas they also considered time windows constraints in their second publication. Their proposed heuristics are especially tailored for the problems and this kind of network structure. Mullaseril et al. (1997) presented a heuristic for the Split-Delivery Capacitated Rural Postman Problem with Time Windows on arcs. The heuristic is similar to the one proposed by Dror and Trudeau (1989, 1990), and is applied to a real-life problem of managing the trucks for distributing feed in a cattle ranch in Arizona. Belenguer et al. (2000) studied the polyhedron of the VRPSD and together with a new class of valid inequalities, a lower bound for the problem is developed.

In this paper, we develop a Tabu Search heuristic for the VRPTWSD where our strategy is different from Dror and Trudeau's (1990). Their work is a two-stage algorithm, where the first stage constructs a VRP solution using node interchanges, and the second stage improves the VRP solution by introducing splits and eliminating splits. Our algorithm does not separate this from one another, it looks at all this in parallel. We have a pool of solutions that are defined by different move operators, the heuristic does not specifically choose to do node interchanges or split the deliveries. That is left to be decided by the heuristic according to the pool of solutions. The best solution in the current pool is always chosen. It could be a split delivery, split-elimination or just a simple node interchange.

The rest of this paper is organized as follows: In Section 2 we give the formulation of the problem, some definitions and notation, and a theoretical result on which our solution method relies. The Tabu Search heuristic based on the four different neighborhood structures is described in Section 3, whereas experimental results are given in Section 4. Finally, a conclusion is drawn in Section 5.

2 Problem Formulation and Definitions

In this section we define the problem under study, and the notation used throughout the paper.

Customers: The problem is given by a set of customers $C = \{1, 2, \ldots, n\}$, residing at n different locations. Letting 0 denote the location of the depot, $N = C \cup \{0\}$ becomes the set of all locations considered in the problem. Every pair of locations (i, j), where $i, j \in N$ and $i \neq j$, is associated with a cost of traveling d_{ij} and a travel time t_{ij}. Every customer $i \in C$ has a demand $w_i > 0$.

Vehicles: A set, V, of vehicles with identical capacities, m, is given. The vehicle set is later referred to as the *fleet*.

Time windows: Each customer i has a *time window*, i.e. an interval $[a_i, b_i] \subseteq \Re$, where a_i and b_i are the earliest and latest time to start to service customer i. A vehicle may arrive at customer location i before a_i, but cannot start servicing until the time window opens at a_i. A vehicle cannot arrive at customer location i after the time window closes at b_i. The depot also has a time window $[a_0, b_0]$, where a_0 represents the earliest time when the vehicles can leave the depot, and b_0 represents the latest time when the vehicles must return to the depot.

Split deliveries: The demand of a customer may be fulfilled by more than one vehicle. This can occur for instance when the demand required by a customer is larger than the capacity of a vehicle, or when it is cheaper to service a customer more than once.

The model contains three sets of decision variables x, f and s. For each arc (i, j), where $i, j \in N$, $i \neq j$ (with the exception of $i = j = 0$ meaning a vehicle is driving an "empty" route), and for each vehicle k, we define x_{ijk} as

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ travels directly from customer } i \text{ to customer } j \\ 0 & \text{otherwise} \end{cases}$$

The decision variable f_{ik} is defined for each customer i and each vehicle k and denotes the fraction of demand of customer i delivered by vehicle k. The last decision variable s_{ik} denotes the time vehicle k starts to service customer i. We assume $s_{0k} = a_0$, for all k.

The VRPTWSD can be stated mathematically[1] as:

$$\min\ z(x) = \sum_{k\in V}\sum_{i\in N}\sum_{j\in N} d_{ij}x_{ijk} \tag{1}$$

$$\text{subject to} \sum_{j\in N} x_{0jk} = 1 \qquad \forall k \in V \tag{2}$$

$$\sum_{i\in N} x_{ihk} - \sum_{j\in N} x_{hjk} = 0 \qquad \forall h \in C, \forall k \in V \tag{3}$$

$$\sum_{k\in V} f_{ik} = 1 \qquad \forall i \in C \tag{4}$$

$$\sum_{i\in C} w_i f_{ik} \leq m \qquad \forall k \in V \tag{5}$$

$$\sum_{j\in N} x_{jik} \geq f_{ik} \qquad \forall i \in C, \forall k \in V \tag{6}$$

$$s_{ik} + t_{ij} - K_{ij}(1 - x_{ijk}) \leq s_{jk} \qquad \forall j \in C, \forall i \in N, \forall k \in V \tag{7}$$

$$a_i \leq s_{ik} \leq b_i \qquad \forall i \in N, \forall k \in V \tag{8}$$

$$s_{ik} + t_{i0} - K_{i0}(1 - x_{i0k}) \leq b_0 \qquad \forall i \in C, \forall k \in V \tag{9}$$

$$f_{ik} \geq 0 \qquad \forall i \in C, \forall k \in V \tag{10}$$

$$s_{0k} = a_0 \qquad \forall k \in V \tag{11}$$

$$x_{iik} = 0 \qquad \forall i \in C, \forall k \in V \tag{12}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j \in N, \forall k \in V \tag{13}$$

The objective (1) is to minimize total distance traveled. The next two sets of equations (2) and (3) state that each vehicle leaves the depot, after arriving at a customer the vehicle leaves again, and it will finally return to the depot. The equation set (4) ensures that every customer receives their full demand, and (5) states that no vehicle is assigned more demand than its capacity. Constraint set (6) ensures that a customer can only be serviced by a vehicle which visits that customer. The inequalities (7) state that a vehicle k cannot arrive at customer j before $s_{ik} + t_{ij}$ if it is traveling directly from customer i to customer j. To this end, the constant K_{ij} is defined sufficiently large, e.g. $K_{ij} = b_i + t_{ij} - a_j$. Constraint set (8) makes sure that every customer is serviced within their time window. Finally, constraint set (9) forces every vehicle to arrive at the depot before the depot's time window closes.

Note that summing up constraint set (6) over all vehicles yields in combination with (4) the

[1] The formulation is based on Frizzell and Giffin (1995) and Larsen (1999).

following constraints:

$$\sum_{k\in V}\sum_{i\in N} x_{ijk} \geq 1 \qquad \forall j \in C \tag{14}$$

implying that every customer receives at least one delivery. The following equation set stating that all vehicles have to return to the depot is omitted from the model

$$\sum_{i\in N} x_{i0k} = 1 \qquad \forall k \in V \tag{15}$$

because it is implied by equation sets (2) and (3).

2.1 Property of VRPTWSD solutions

For a given feasible solution $(x, f, s) \in B^{|N|\times|N|\times v} \times \Re_+^{|N|\times v} \times \Re_+^{|N|\times v}$ to VRPTWSD, each $k \in V$ defines a unique ordered set $R_k = (0, c_1^k, c_2^k, \ldots, c_g^k, 0)$ such that $x_{0c_1^k k} = x_{c_1^k c_2^k k} = \ldots = x_{c_g^k 0k} = 1$. We shall refer to R_k as the *route* of vehicle k (induced by x). For notational convenience, we shall in the remainder of the paper assume that $c_i^k = i$, although this generally is not the case. Hence i refers to the position of the customer in R_k, and $i-1$ and $i+1$ are respectively the predecessor and the successor of i in this particular route.

The following property is proved by Dror and Trudeau (1989, 1990) in the case of VRPSD, and we will prove that it also applies to VRPTWSD.

Proposition 1 *If VRPTWSD is feasible and $\{t_{ij}\}_{i,j\in N}$ satisfy the triangle inequality, then the problem has an optimal solution where no two of the induced routes have more than one common customer.*

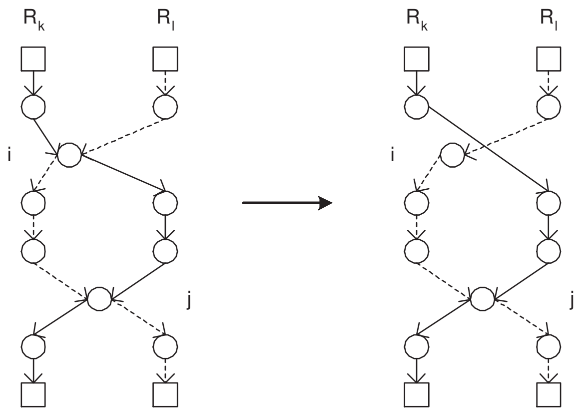

Figure 1: The two-route two-split example: *two-split* (left) and *one-split* (right). Squares represent the depot (which is duplicated at each end) and circles represent customers in the route.

Proof Assume that $(\hat{x}, \hat{f}, \hat{s})$ is an optimal solution to VRPTWSD, and let $\hat{R}_1, \ldots, \hat{R}_v$ be the routes induced by $\hat{x}$. Also assume that $i, j \in \hat{R}_k \cap \hat{R}_l$ where $i, j \in C$ and $1 \leq k < l \leq v$, i.e. the deliveries at both i and j are split between vehicles k and l (see Figure 1, left). We shall construct a feasible solution (x, f, s) with induced routes $R_1, \ldots, R_v$ such that $z(x) \leq z(\hat{x})$, $i \notin R_k \cap R_l$, and such that $R_k \cap R_l \subset \hat{R}_k \cap \hat{R}_l$ (e.g. Figure 1, right). If such a solution can be found, the proposition is proven.

We let $\hat{q}_{ik} = \hat{f}_{ik} w_i$ denote the quantity delivered to customer i by $\hat{R}_k$. Without loss of generality, it can be assumed that $\hat{q}_{ik} = \min\{\hat{q}_{ik}, \hat{q}_{il}, \hat{q}_{jk}, \hat{q}_{jl}\}$. Define q equal to $\hat{q}$ and x equal to $\hat{x}$ in all components with the following exceptions: $q_{jk} = \hat{q}_{jk} + \hat{q}_{ik}$, $q_{jl} = \hat{q}_{jl} - \hat{q}_{ik}$, $q_{il} = \hat{q}_{il} + \hat{q}_{ik}$, $q_{ik} = 0$, $x_{i-1,i+1,k} = 1$, $x_{i-1,i,k} = 0$ and $x_{i,i+1,k} = 0$. Letting $f_{ik} = q_{ik}/w_i$ for all $(i,k) \in C \times V$, it follows from Dror and Trudeau (1989, 1990) that (x, f) satisfies (2)-(6), (10) and (12)-(13), and that $z(x) \leq z(\hat{x})$. Since $\hat{s}$ satisfies (8) and (11), putting $s = \hat{s}$ ensures that s satisfies these constraints. It remains to show that this choice of s also satisfies (7) and (9). But the only affected constraints are those indexed by $(i-1, i, k)$, $(i, i+1, k)$ and $(i-1, i+1, k)$. The first two are trivially satisfied since $x_{i-1,i,k} = x_{i,i+1,k} = 0$. If $i + 1 \in C$, we have $s_{i-1,k} + t_{i-1,i} \leq s_{ik}$ and $s_{ik} + t_{i,i+1} \leq s_{i+1,k}$, since $(\hat{s}, \hat{x})$ satisfies (7) and $\hat{x}_{i-1,i,k} = \hat{x}_{i,i+1,k} = 1$. Since $t_{i-1,i+1} \leq t_{i-1,i} + t_{i,i+1}$, we thus get $s_{i-1,k} + t_{i-1,i+1} \leq s_{i-1,k} + t_{i-1,i} + t_{i,i+1} \leq s_{ik} + t_{i,i+1} \leq s_{i+1,k}$. Since $x_{i-1,i+1,k} = 1$, we have $s_{i-1,k} + t_{i-1,i+1} - K_{i-1,i+1}(1 - x_{i-1,i+1,k}) \leq s_{i+1,k}$, and (7) is satisfied. If $i + 1$ is the depot, a similar argument is used in order to prove that (9) is satisfied.

3 Tabu Search

Tabu Search is a memory-based search strategy, originally proposed by Glover (see Glover and Laguna 1997), to guide the local search method to continue its search beyond a local optimum. One way of achieving this is to keep track of recent moves or solutions made in the past. Tabu list records recently made moves or visited solutions. Whenever the algorithm attempts to make a move listed in the tabu list, the move is banned. By this, the algorithm forces other solutions to be explored. However, this feature is not strict, it can be overridden when some aspiration criteria are satisfied. A popular aspiration criterion is that the target function value be the best ever seen. If this is the case, it is obvious that this solution has never been encountered before. This is the reason for accepting the solution, although it is forbidden by the tabu list.

A solution to the problem expressed by (1)-(13) is given by a set of routes, an assignment of customers to routes, and a service start time and delivery fraction for each customer on each route. We let $\sigma = \{R_1, \ldots, R_v\}$ denote the set of routes. Let S be the set of route sets for which feasible service start times and delivery fractions exist. Each solution $\sigma \in S$ has an associated set of neighbors, $N(\sigma) \subset S$, called the *neighborhood* of σ. Each solution $\sigma' \in N(\sigma)$ can be reached directly from σ by a *move*. A move is a transition from σ to σ' by means of a move operator described in Section 3.2.

Our solution method is a three-step process. First we compute an initial feasible solution by simple consideration of the combination of travel time and waiting time. Next we try to improve it by Tabu Search, and finally a post-optimization phase is applied to it.

3.1 Initial solution

The initial solution is computed by appending the nearest unrouted neighbor $\hat{j}$ to customer i (the latest routed customer) with respect to minimum sum of travel time and waiting time from i to $\hat{j}$. This process repeats until all customers are routed, as described in Algorithm 1, where θ_i is the time the vehicle starts to service customer i, and

$$\hat{j} \in \arg\min\{t_{ij} + \max\{a_j - \theta_i - t_{ij}, 0\} | j \in C\} \tag{16}$$

is the criterion for choosing $\hat{j}$. Notice that if neighbor $\hat{j}$ is picked and its demand exceeds the vehicle's capacity, $\hat{j}$ is still appended and the excess demand is left to some other vehicle(s) to handle. In this way an initial solution with split deliveries is made. Algorithm 1 does not determine whether the split is profitable or not.

Note that Algorithm 1 may end up with more routes than the actual number of vehicles available.

Algorithm 1 Initial Solution

$k = 1$
repeat
  Begin with an empty route k starting from the depot, $R_k = (0, 0)$.
  Set $\hat{j} = 0$ and $\theta_0 = a_0$.
  repeat
    Set $i = \hat{j}$.
    Find a nearest unrouted (i.e. $\sum_{\bar{k} \in V} f_{\hat{j}\bar{k}} < 1$) neighbor $\hat{j}$ to customer i w.r.t. (16) and feasibility in terms of time constraints.
    Insert $\hat{j}$ in the current route k, and set $\theta_{\hat{j}} = \theta_i + t_{i\hat{j}} + \max\{a_{\hat{j}} - \theta_i - t_{i\hat{j}}, 0\}$.
    Compute the spare capacity u_k.
  until $u_k = 0$ or no more insertions are valid
  $k = k + 1$
until all customers are routed

If the succeeding solution method is unable to reduce the number of routes to v or less, it reports that no feasible solution is found.

We will now review some features used in the Tabu Search algorithm.

3.2 Neighborhoods

The neighborhood of our tabu search algorithm is based on some common move operators; *relocate*, *exchange* and *2-opt** (these are modified a little for handling split deliveries) and one move operator called *relocate split*. Following is a short description of each one of them.

1. **Relocate operator** (see Figure 2): For locations $i \in R_k \cap C$ and $j_\alpha \in R_{l_\alpha}$, place i after j_α in R_{l_α}, where $\alpha = 1, \ldots, \beta$, and β is the number of necessary vehicles servicing i with $\beta \geq 1$. The new vehicle routes are $R'_k = (0, \ldots, i-1, i+1, \ldots, 0)$ and $R'_{l_\alpha} = (0, \ldots, j_\alpha, i, j_\alpha + 1, \ldots, 0)$. In cases where $i \in R_k \cap R_{l_\alpha} \cap C$, a split delivery may be eliminated, and the position of i in R_{l_α} remains unchanged.

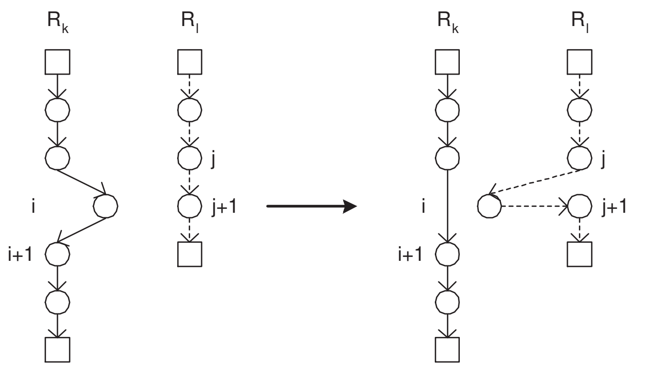

Figure 2: A relocate operation

2. **Relocate split operator** (see Figure 3): For customers $i \in R_k \cap R_l \cap C$ and $j \in R_l \cap C$: Remove customer i from R_k, and let $\hat{R}_k$ be $(0, \ldots, i-1, i+1, \ldots, 0)$. While j is serviced by R_l, let j also be serviced by $\hat{R}_k$. The new vehicle routes are $R'_l = R_l$ (but with a higher quantity delivered to i and less delivered to j) and $R'_k = \hat{R}_k \cup \{j\}$.

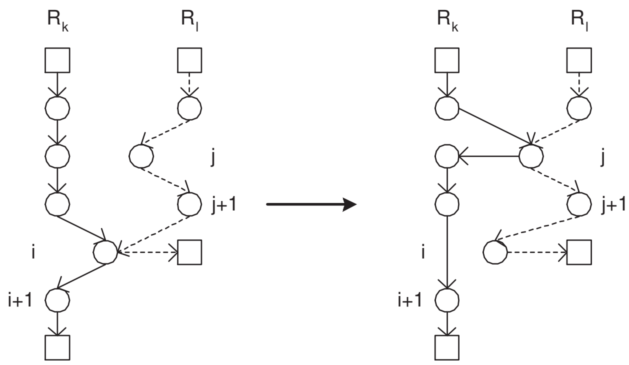

Figure 3: A relocate split operation

3. **Exchange operator** (see Figure 4): The exchange is between a pair of routes R_k and R_l, and is an exchange of customers $i \in R_k \cap C$ and $j \in R_l \cap C$. Customer i is inserted in $R_l \setminus \{j\}$, but not necessarily in the position held by j. Similarly, j is inserted in some position in $R_k \setminus \{i\}$. In cases where $i \in R_k \cap R_l \cap C$, $i \neq j$, a split delivery may be eliminated.

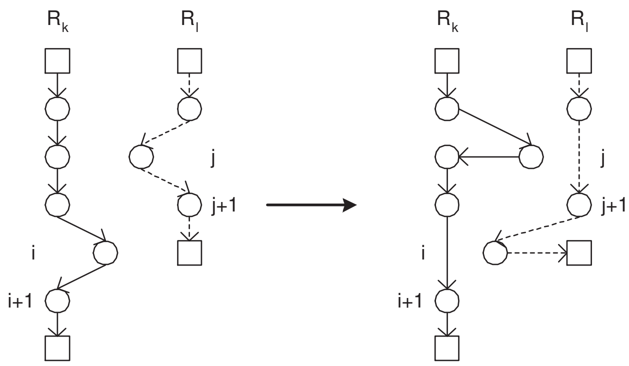

Figure 4: An exchange operation

4. **2-opt* operator** (see Figure 5): For locations $i \in R_k$ and $j \in R_l$, let the new vehicle routes be $R'_k = (0, \ldots, i, j+1, j+2, \ldots, 0)$ and $R'_l = (0, \ldots, j, i+1, i+2, \ldots, 0)$. In cases where $i \in R_k \cap R_l \cap C$ and i coincides with $j+1$, a split delivery between R_k and R_l may be eliminated.

The neighborhoods defined by the above operators are denoted N_1, N_2, N_3 and N_4, respectively.

Some other phases are also used in the heuristic:

Route saving phase: In this phase we only consider vehicle routes with three or less customers. The heuristic tries to insert each of the route's customer into the other routes using the relocate operator. If the routes can successfully be emptied, and still obtain distance savings, then these routes are removed from the solution, otherwise the heuristic keeps the original solution.

US: This is an improvement phase that is a part of the GENIUS algorithm developed by Gendreau et al. (1998). The US algorithm removes each customer in turn (starting with the first customer of the route), and reinserts it while performing a local reoptimization of the route. Both the reinsertion and the reoptimization part have two different types of unstringing/stringing processes, but we only consider type 1 unstringing/stringing. Type 1 insertions (i.e. stringing) are equivalent

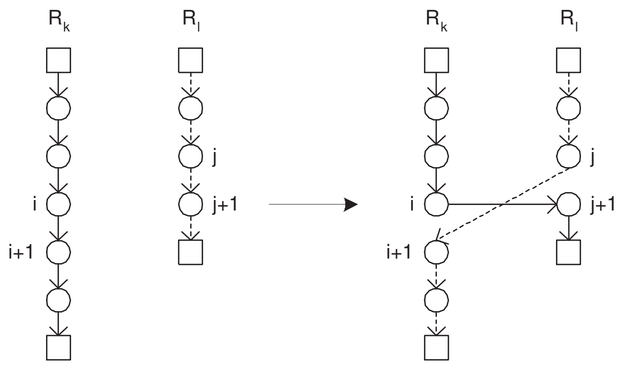

Figure 5: A 2-opt* operation

to selecting the best of several moves, each consisting of a simple insertion followed by only one 3-opt exchange. Type 1 unstringing refers to removing a customer from a route, and rearrange the rest of the customers of the same route in a different way (using something similar to 3-opt). Details can be found in Gendreau et al. (1992, 1998).

3.3 Tabu Search for VRPTWSD

At each iteration the four different move operators are applied to the current solution. The best feasible neighbor solution found is chosen. The solution is either tabu or non-tabu. It is tabu if the move that lead to the solution is tabu, but the tabu status may be overridden by an aspiration criterion (which is the best total cost seen so far). The inverse move is then set tabu for the next p iterations, and will not be classified as a valid move until the tabu status is expired or is overridden by the aspiration criterion.

For any feasible route set σ, let $z(\sigma)$ be the travel cost. The move operators *relocate*, *relocate split*, *exchange* and *2-opt** are associated with $1, \ldots, 4$ respectively. We let $T_i(\sigma)$ be the set of solutions to which the move defined by operator i is tabu, where $i = 1, \ldots, 4$. We also define $A_i(\sigma) \subseteq T_i(\sigma)$ to be the set of solutions with tabu moves overridden by the aspiration criterion.

What kind of moves are considered tabu? Assume we have routes R_k and R_l where $i \in R_k$ and $j \in R_l$, by applying the different move operators we obtain the following tabu moves:

1. **Relocate operator**: We place $i \in R_k \cap C$ after j_α in R_{l_α}. Then the tabu list records the tabu status of customer i in R_k, after being moved to R_{l_α}. That is, we set $TABU(i, R_k) = p$, meaning moving $i \in R_{l_\alpha}$ to R_k is tabu for the next p iterations. We let $T_1(\sigma)$ consist of all solution in $N_1(\sigma)$ where $i \in R_k$ and $TABU(i, R_k) > 0$ for some i and k.

2. **Relocate split operator**: We assume that $i \in R_k \cap R_l \cap C$, and after applying the move operator, customer $j \in R_k \cap R_l \cap C$ and $i \notin R_k$. The tabu list records the tabu status of customer i in R_k, after being moved to R_l. That is, we set $TABU(i, R_k) = p$, meaning moving i to R_k is tabu for the next p iterations. We let $T_2(\sigma)$ be defined in a way analogous to $T_1(\sigma)$.

3. **Exchange operator**: We exchange customer i with customer j. The tabu list records the tabu status of customer i in R_k, after being moved to R_l, and it also records the tabu status of customer j in R_l, after being moved to R_k. That is, we set $TABU(i, R_k) = p$ and $TABU(j, R_l) = p$, meaning moving i to R_k and moving j to R_l are tabu for the next p iterations. We let $T_3(\sigma)$ be defined in a way analogous to $T_1(\sigma)$.

4. **2-opt* operator**: We exchange i's descendants with j's descendants. The tabu list records the tabu status of i and $i+1$, and of j and $j+1$. That is, we set $TABU(i, i+1) = p$ and $TABU(j, j+1) = p$, meaning the arcs $(i, i+1)$ and $(j, j+1)$ are forbidden to remove for the next p iterations. We let $T_4(\sigma)$ consist of all solutions in $N_4(\sigma)$ where h is the successor of i in some route, and $TABU(i, h) > 0$.

Note that $T_4(\sigma)$ differs in notation from the first three tabu lists. The second entry is a customer and no longer a route. In other words, the tabu list for the first three operators deals with the vertices (i.e. customers), while the second tabu list deals with the arcs.

The route saving phase is performed once in every q iterations. The heuristic ends when y consecutive iterations are performed without any improvement to the best known solution. Finally, a post-optimization phase is applied to the best solution found in the search.

Thus, the heuristic can be summarized as in Algorithm 2.

Algorithm 2 Tabu Search heuristic

Obtain an initial solution by Algorithm 1, followed by US and a route saving phase, and denotes this solution σ. Set $\sigma^* = \sigma$. Initiate tabu lists.
repeat
  for $i = 1, \ldots, 4$ **do**
    $\sigma'_i = \arg\min\{z(\gamma) | \gamma \in (N_i(\sigma) - T_i(\sigma)) \cup A_i(\sigma)\}$
  i' =argmin$\{z(\sigma'_i) | i = 1, \ldots, 4\}$
  $\sigma = \sigma'_{i'}$
  Set the inverse move of σ tabu for the next p iterations, and decrement previous entries in the tabu lists by one.
  if $z(\sigma) < z(\sigma^*)$ **then**
    $\sigma^* = \sigma$
  Perform route saving phase once in q iterations.
until y consecutive iterations are performed without any improvement to the best known solution
Apply post-optimization phase (US) on the best solution.

3.4 VRPTW version of the heuristic

We have also developed a VRPTW version of the heuristic proposed in the previous subsection. That is, we make changes to Algorithms 1 and 2 necessary to produce a feasible solution to VRPTW, where no split deliveries are allowed. Concerning the initial solution, Algorithm 1 shows the VRPTWSD version where a customer is appended to the route without determining if the vehicle is capable to handle the whole demand, only whether it is full-loaded or not. In the VRPTW version, the vehicle's spare capacity must be at least as large as the customer's demand in order to let the customer be serviced by this vehicle.

There are also a few minor adaptions concerning the neighborhood structures. In the case of the relocate operator, customer i can only be moved to one vehicle, contrary to Section 3.2 where customer i is allowed to be transferred to more vehicles. Since we do not allow any split deliveries to be made, the relocate split operator is never used (because its precondition is a split delivery between two routes). The other two move operators remain unchanged.

The final adaption is made in the route saving phase, which is due to the relocate operator mentioned above. These are all the changes needed to be incorporated in the Tabu Search heuristic in order to have a VRPTW version of the heuristic.

4 Experimental results

4.1 Problem sets

Standard test cases for the VRPTW are used for experimentation. The Solomon test problems consist of 100 customers with Euclidean distance. In these problems, the travel times are equal to the corresponding Euclidean distances. We consider the travel times as a composition of the corresponding Euclidean distances and service time. There are six sets of problems where the geographical data are either randomly generated according to a uniform distribution (problem sets R1 and R2), clustered in groups (problem sets C1 and C2), or semiclustered (problem sets RC1 and RC2). By a semiclustered problem, we mean one that contains a mix of randomly generated data and clusters. Further, problem sets R1, C1 and RC1 have a short scheduling horizon, and combined with a constrained vehicle capacity, it allows only a few (3-8) customers to be serviced by the same vehicle. The sets R2, C2 and RC2 have a long scheduling horizon, and larger vehicle capacity yielding routes with more (i.e. 10+) customers. A thorough description of the test sets is given in Solomon (1987).

The computational experiments also include modified Solomon test problems which are described in Subsection 4.4.

4.2 Comparison with best published VRPTW results

In this subsection we compare the VRPTW version of our heuristic with other heuristics found in the literature. Several benchmarks for this problem exist, and several authors compete to publish the best solutions. A sample of them are (abbreviations have been used in Tables 2 and 3): BV=Bent and Van Hentenryck (2001), CLM=Cordeau et al. (2001), DFSKP=De Backer et al. (2000), GTA=Gambardella et al. (1999), HG=Homberger and Gehring (1999), RGP=Rousseau et al. (2002), RT=Rochat and Taillard (1995), S=Shaw (1998) and TBGGP=Taillard et al. (1997).

The results in this and the following subsections were obtained with the parameter settings presented in Table 1. Note that the parameter settings were chosen in an arbitrary fashion and we have not tuned their values. The heuristic described in the previous section is coded in C++, compiled using Sun C++ compiler and run on a Sun Ultra 10 (UltraSPARC-IIi 440 MHz) workstation. Tables 2 and 3 show the comparison between the best published results and the results from our heuristic. The column labeled *Best* shows the total distance traveled and the number of vehicles utilized. *Ref.* gives references to where the best published solutions can be found. The columns labeled *VRPTW* and *VRPTWSD* report the results obtained by the VRPTW and VRPTWSD versions of our heuristic. Although VRPTWSD is an interesting problem in its own right, it can also be considered as a relaxation of VRPTW. Hence, our VRPTWSD heuristic produces solutions to a relaxation of VRPTW, and in cases where the split delivery option is not taken advantage of, the relaxed solution is actually a feasible VRPTW-solution. This turned out to be true for all experiments reported in Tables 2 and 3.

The boldfaced results indicate matchings of the best results, while italized results represent improvements over the best known solutions.

Table 1: Parameter settings for the Heuristic

Tabu tenure: $p = 15$
Stopping criterion: $y = 100$
Performing route saving phase criterion: $q = 5$
Parameters used in Gendreau et al. (1998): $p_1 = 4$, $p_2 = 5$

Table 2: Comparison between the best published results of Solomon's test cases (class 1) and ours

Problem	Best		Ref.	VRPTW		VRPTWSD	
R101	1650.8	19	RT	*1648.96*	20	*1648.96*	20
R102	1486.12	17	RT	1513.57	18	1506.61	18
R103	1292.68	13	S	*1281.14*	15	*1278.93*	15
R104	1007.31	9	S	1099.93	13	1099.93	13
R105	1377.11	14	RT	1453.58	16	1453.58	16
R106	1252.03	12	RT	1292.29	14	1292.29	14
R107	1104.66	10	S	1154.04	12	1150.06	12
R108	960.876	9	BV	979.52	10	1004.42	11
R109	1194.73	11	HG	1270.66	13	1270.66	13
R110	1118.84	10	BV	1232.53	13	1232.53	13
R111	1096.72	10	RGP	1142.47	13	1106.37	12
R112	982.14	9	GTA	1049.98	11	1042.44	11
C101	828.94	10	RT	**828.94**	**10**	**828.94**	**10**
C102	828.94	10	RT	853.39	**10**	842.02	**10**
C103	828.06	10	RT	878.70	**10**	890.57	**10**
C104	824.78	10	RT	896.40	**10**	894.41	**10**
C105	828.94	10	RT	**828.94**	**10**	**828.94**	**10**
C106	828.94	10	RT	**828.94**	**10**	**828.94**	**10**
C107	828.94	10	RT	**828.94**	**10**	**828.94**	**10**
C108	828.94	10	RT	830.33	**10**	830.33	**10**
C109	828.94	10	RT	830.33	**10**	830.33	**10**
RC101	1696.94	14	TBGGP	1748.15	16	1748.15	16
RC102	1554.75	12	TBGGP	1568.86	15	1568.86	15
RC103	1261.67	11	S	1325.84	12	1325.84	12
RC104	1135.48	10	S	1222.79	11	1222.79	11
RC105	1629.44	13	BV	1655.93	17	1655.93	17
RC106	1424.73	11	BV	1443.88	14	1443.88	14
RC107	1230.48	11	S	1308.99	12	1278.21	12
RC108	1139.82	10	TBGGP	1332.32	12	1314.01	12

Table 3: Comparison between the best published results of Solomon's test cases (class 2) and ours

Problem	Best		Ref.	VRPTW		VRPTWSD	
R201	1252.37	4	HG	1271.17	5	1271.17	5
R202	1191.7	3	RGP	*1171.38*	4	*1171.38*	4
R203	941.408	3	BV	1034.94	4	1034.94	4
R204	825.519	2	BV	854.85	4	854.85	4
R205	994.42	3	RGP	1078.79	4	1078.79	4
R206	912.97	3	RT	1010.33	**3**	1010.33	**3**
R207	893.328	2	BV	1009.08	3	1009.08	3
R208	726.823	2	GTA	827.12	3	827.12	3
R209	909.163	3	BV	1046.69	**3**	1046.69	**3**
R210	939.37	3	DFSKP	1015.93	4	1015.93	4
R211	892.713	2	BV	*839.85*	3	*839.85*	3
C201	591.56	3	RT	**591.56**	**3**	**591.56**	**3**
C202	591.56	3	RT	620.29	**3**	620.29	**3**
C203	591.17	3	RT	623.24	**3**	623.24	**3**
C204	590.60	3	RT	645.26	**3**	645.26	**3**
C205	588.88	3	RT	596.89	**3**	596.89	**3**
C206	588.49	3	RT	625.71	**3**	625.71	**3**
C207	588.29	3	RT	597.25	**3**	597.25	**3**
C208	588.32	3	RT	613.96	**3**	613.96	**3**
RC201	1406.94	4	CLM	*1379.53*	5	*1379.53*	5
RC202	1377.089	3	GTA	1390.44	4	1390.44	4
RC203	1060.45	3	HG	1150.76	4	1150.76	4
RC204	798.464	3	GTA	1021.6	**3**	1021.6	**3**
RC205	1297.65	4	BV	1482.17	5	1482.17	5
RC206	1146.32	3	BV	1260.99	4	1260.99	4
RC207	1061.14	3	BV	1107.84	4	1107.84	4
RC208	828.141	3	BV	1029.76	**3**	1029.76	**3**

Tables 2 and 3 show that our heuristic has improved (in terms of total distance traveled) 5 (8.93%) of the 56 best published solutions to the Solomon benchmarks, while matching or improving the best solutions in 10 problems (17.86%). However, our solutions utilize more vehicles than the best published in almost two thirds of the instances.

4.3 Discussion

The demand by vehicle capacity ratios, or demand ratios for short, for the original Solomon benchmarks are the following:

R1:	0.005 - 0.205	C1:	0.05 - 0.25	RC1:	0.01 - 0.2
R2:	0.001 - 0.041	C2:	0.014 - 0.071	RC2:	0.002 - 0.04

The ratios are very small and together with a short scheduling horizon it is most unlikely that any split deliveries will be made. This is confirmed by the experiments reported in Tables 2 and 3. From Table 2 we can see that most of the VRPTWSD solutions coincide with the VRPTW solutions. There are two instances where the VRPTW version of the heuristc does give better solutions than VRPTWSD, and there are eight instances where VRPTWSD is better than VRPTW.

Why is it that VRPTW is better than VRPTWSD for two of the instances? This can be explained by different initial solutions for the two different versions of the problem. One initial solution contains split deliveries, while the other does not. Other factors are the relocate split operator, split deliveries and spare capacities, all these combined will give different search trajectories. We have also conducted experiments that used the VRPTW version of the heuristic to generate the initial solution, and the VRPTWSD version doing the search. All results coincide with the VRPTW results. The only exceptions are two instances that yielded better results than the VRPTW version.

Table 4: Mean values of results

	R1	R2	C1	C2	RC1	RC2
VRPTW	14	3.64	10	3	13.63	4
	1259.89	1014.56	844.99	614.27	1450.76	1227.89
VRPTWSD	14	3.64	10	3	13.63	4
	1257.23	1014.56	844.82	614.27	1444.71	1227.89

Table 4 shows the mean values of the number of vehicles utilized and total distance traveled. As reported in this table, VRPTWSD is doing slightly better than VRPTW, even without taking advantage of its splitting option. All split deliveries that occurred in the initial solutions are eliminated during the search. The reported findings in the previous subsection have shown that our heuristic does give better solutions for the VRPTWSD than for VRPTW, when the demand ratios are low. As demonstrated in the next subsection, this effect is amplified as the demand ratio grows.

4.4 Experiments on Solomon's problems with augmented demands

We have slightly modified Solomon's test instances in order to study how the value of the splitting option varies with the demand ratio. All demands are transformed affinely to the interval $[lm, um]$, where $l < u$ are given numbers in the interval $(0, 1]$. That is, for all $i \in C$ we define the new demand $w_i' = lm + m\frac{u-l}{\bar{w}-\underline{w}}(w_i - \underline{w})$ where $\underline{w} = \min\{w_i : i \in C\}$ and $\bar{w} = \max\{w_i : i \in C\}$. Results from experiments with Solomon's problems for various values of l and u are reported in Table 5.

Table 5 shows the mean values of the number of vehicles utilized and total distance traveled of the VRPTW and VRPTWSD version of the heuristic. To conduct this experiment we let the initial solutions be the same (i.e. we do not allow split deliveries in the initial solution) for both of the versions of the heuristic. We choose to do this because of the larger demands and the larger neighborhood for the VRPTWSD (as a result from high demand ratios). Since the demand ratios are higher, also the variations between VRPTW and VRPTWSD are bigger when choosing the most profitable move at each iteration. As reported in Table 5, the overall results favor VRPTWSD, both in terms of total distance traveled and number of vehicles utilized.

Table 5: Comparison of results for the modified problem instances from Solomon

l, u		R1	R2	C1
0.01,0.50	VRPTW	18.67/1483.76	18.00/1420.43	13.30/1104.20
	VRPTWSD	18.25/1473.92	18.00/1423.53	12.78/1098.79
0.02,1.00	VRPTW	36.92/2329.85	37.00/2333.40	27.00/1999.80
	VRPTWSD	35.17/2316.91	35.00/2318.59	23.11/1838.60
0.50,1.00	VRPTW	99.08/4962.56	99.00/4960.12	74.00/4631.12
	VRPTWSD	67.83/3934.86	68.39/3966.13	61.00/3966.67
0.70,1.00	VRPTW	100.00/4989.42	100.00/4989.42	100.00/5770.96
	VRPTWSD	80.50/4463.08	81.18/4496.28	77.22/4929.52

l, u		C2	RC1	RC2
0.01,0.50	VRPTW	12.25/1226.58	21.13/1945.52	20.38/1937.73
	VRPTWSD	12.00/1152.41	20.50/1939.31	20.00/1953.16
0.02,1.00	VRPTW	27.00/2018.50	42.63/3421.45	41.75/3384.32
	VRPTWSD	23.00/1854.40	40.00/3317.92	39.75/3301.14
0.50,1.00	VRPTW	74.00/4722.74	100.00/6617.54	100.00/6617.54
	VRPTWSD	61.00/4180.18	70.25/5335.26	71.63/5401.52
0.70,1.00	VRPTW	100.00/5942.81	100.00/6617.54	100.00/6617.54
	VRPTWSD	77.13/5155.08	81.88/5998.34	83.00/6052.21

Only for problem class R2 and RC2 for values of 0.01 and 0.5 of l and u, the VRPTW version is better than the VRPTWSD version. The biggest savings, both in terms of distance and the number of vehicles are yielded in the last two categories. When the demand ratios are at least 0.5, there are not many combinations to arrange a sequence of the customers of the VRPTW version. But for the VRPTWSD version of the heuristic, there are considerable more choices. This is due to the numerous splits generated by the route saving phase, relocate and relocate split operators. The number of splits decreased as the demand by vehicle capacity ratio decreased.

Total customer visits are $n = 100$ for all problem instances of the VRPTW since each customer can be serviced only once. For VRPTWSD, this figure is the sum of the degrees of all deliveries (an unsplit delivery has degree 1, a delivery split between β vehicles has degree β). The total number of visits tends to grow as the demand ratio increases. Table 6 shows average customer visits for the six problem classes (with 1=R1, 2=R2, 3=C1, 4=C2, 5=RC1 and 6=RC2) and various values of l and u. This growth is due to the reduction of vehicles obtained by applying the relocate and relocate split operators. When the respective demands occupy much of the vehicles, applying the relocate operator will result in numerous splits for the customers. As demand ratios grow, the heuristic tends to operate with the relocate split operator more often than when the ratios are small. This operator will also generate splits in certain cases.

Table 7 shows the mean values of the distribution of moves of various values of l and u for

Table 6: Average customer visits

l, u	R1	R2	C1	C2	RC1	RC2
0.01,0.50	103.00	104.27	100.78	102.25	103.88	106.13
0.02,1.00	117.08	116.18	112.33	113.38	119.75	121.13
0.50,1.00	159.67	163.09	125.56	132.63	158.25	159.75
0.70,1.00	168.42	169.45	167.44	173.63	168.75	168.00

the VRPTWSD. For the first category we see that the relocate operator clearly dominates the distribution. When the demand ratios are relatively small, it is profitable to move customers from one route to another using relocate operator. The relocate split operator takes over as the dominator for the other values of l and u (except for problem classes C1 and C2 of 0.2 and 1). The reason for this can be found in the constrained spare capacities and the relatively small demand ratios. When applying the relocate operator, it will result in numerous splits and will not be profitable. A better alternative is the relocate split operator, although in certain cases a split will be made.

Table 7: Average distribution of moves

l, u	operator	R1	R2	C1	C2	RC1	RC2
0.01,0.50	relocate	174.58	237.18	167.22	200.50	107.13	134.13
	relocate split	65.08	188.55	4.44	15.00	64.38	93.13
	exchange	43.83	73.09	15.89	19.38	43.25	51.88
	2-opt*	58.00	72.91	16.00	31.00	42.13	52.25
0.02,1.00	relocate	86.42	109.00	97.89	110.88	60.38	92.50
	relocate split	217.00	324.09	64.00	84.50	157.88	265.38
	exchange	46.00	64.09	32.78	28.63	36.25	67.88
	2-opt*	30.75	41.73	26.11	42.13	28.38	42.13
0.50,1.00	relocate	9.67	7.45	20.56	16.38	13.75	5.38
	relocate split	208.67	177.82	181.67	264.63	217.25	217.50
	exchange	56.33	41.91	157.00	159.88	35.25	23.13
	2-opt*	6.50	6.73	27.22	29.5	11.25	10.38
0.70,1.00	relocate	2.33	2.45	5.89	7.75	5.38	3.25
	relocate split	95.58	94.00	166.44	205.38	124.75	97.25
	exchange	84.17	75.73	51.11	49.88	59.13	61.50
	2-opt*	2.33	3.91	12.22	10.63	9.13	7.50

4.4.1 Computation time

Table 8 shows the mean values of the CPU time the workstation consumed in order to perform the experiments described in Subsection 4.4.

The VRPTW version is faster when the demand ratios are high. Conversely for the VRPTWSD version, it is computationally more expensive to run on problems with high demand ratios. The reason for this lies in the possibility of moving customers between the routes without violating constraints. The set of feasible solutions to VRPTW becomes very restricted as the ratio grows, resulting in a rapid search.

Table 8: Computation time (min:sec) for the modified problem instances from Solomon

l, u		R1	R2	C1	C2	RC1	RC2
0.01,0.50	VRPTW	8:50	11:12	6:29	6:25	3:16	5:07
	VRPTWSD	11:31	27:08	8:25	13:48	8:04	12:28
0.02,1.00	VRPTW	5:48	6:20	4:48	5:07	4:15	5:16
	VRPTWSD	19:18	32:42	12:29	16:51	13:50	25:30
0.50,1.00	VRPTW	4:34	5:00	4:46	4:45	4:54	5:03
	VRPTWSD	27:08	23:10	15:48	25:09	23:02	22:44
0.70,1.00	VRPTW	4:25	4:52	4:52	4:45	4:50	4:55
	VRPTWSD	18:40	18:46	20:58	27:54	18:28	16:30

Table 9 shows the average number of iterations needed to run a test problem of various classes. As depicted in Tables 8 and 9, the iteration number decreased as the demand ratio increased. However, the computation time has not decreased as a result of this, and it rather increased due to more time needed to perform one single iteration. This can be explained by more legal moves due to the splitting option.

Table 9: Iterations needed for the modified problem instances from Solomon

l, u		R1	R2	C1	C2	RC1	RC2
0.01,0.50	VRPTW	286.00	358.73	229.44	192.63	261.63	336.00
	VRPTWSD	339.00	571.73	236.89	265.88	256.88	331.38
0.02,1.00	VRPTW	294.33	298.45	182.11	190.50	212.38	248.25
	VRPTWSD	380.17	538.91	220.78	266.13	282.88	467.88
0.50,1.00	VRPTW	100.67	109.64	152.00	150.88	121.88	127.00
	VRPTWSD	281.17	224.82	386.44	470.38	277.50	256.38
0.70,1.00	VRPTW	100.67	109.64	101.78	100.00	121.88	127.00
	VRPTWSD	184.42	176.09	235.67	273.63	198.38	169.50

5 Conclusions

We have proposed a Tabu Search heuristic for the VRPTWSD. The heuristic minimizes the total distance traveled using Tabu Search with four different neighborhood structures. Results on both the VRPTW and VRPTWSD versions of the heuristic are reported. Experimental results have shown that the heuristic has improved (in terms of total distance) 5 (8.93%) of the 56 best published solutions to the Solomon benchmarks, while matching or improving the best solutions in 10 problems (17.86%). We have also conducted experiments on how the two versions of the heuristic perform to different demand by vehicle capacity ratios. The results have shown that the VRPTWSD version is capable of getting lower distance and number of required vehicles than the VRPTW version. This is especially remarkable for test problems where the demand by vehicle capacity ratios are at least 0.5, but these test instances require considerable more time to run. These problems also resulted in more customer visits.

Acknowledgement

This work is supported by The Research Council of Norway under grant 127533/432.

References

Belenguer, J. M., Martinez, M. C. and Mota, E. (2000), 'A lower bound for the split delivery vehicle routing problem', *Operations Research* **48**(5), 801–810.

Bent, R. and Van Hentenryck, P. (2001), A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows, Technical Report CS-01-06, Brown University, Providence, USA.

Bräysy, O. and Gendreau, M. (2001), Metaheuristics for the Vehicle Routing Problem with Time Windows, Technical Report STF42 A01025, SINTEF Applied Mathematics, Department of Optimization, Oslo, Norway.

Cordeau, J.-F., Laporte, G. and Mercier, A. (2001), 'A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows', *Journal of the Operational Research Society* **52**(8), 928–936.

De Backer, B., Furnon, V., Shaw, P., Kilby, P. and Prosser, P. (2000), 'Solving Vehicle Routing Problems using Constraint Programming and Metaheuristics', *Journal of Heuristics* **6**(4), 501–523.

Dror, M., Laporte, G. and Trudeau, P. (1994), 'Vehicle routing with split deliveries', *Discrete Applied Mathematics* **50**(3), 239–254.

Dror, M. and Trudeau, P. (1989), 'Savings by split delivery routing', *Transportaion Science* **23**(2), 141–149.

Dror, M. and Trudeau, P. (1990), 'Split delivery routing', *Naval Research Logistics* **37**(3), 383–402.

Frizzell, P. W. and Giffin, J. W. (1992), 'The bounded split delivery vehicle routing problem with grid network distances', *Asia-Pacific Journal of Operational Research* **9**(1), 101–116.

Frizzell, P. W. and Giffin, J. W. (1995), 'The split delivery vehicle scheduling problem with time windows and grid network distances', *Computers & Operations Research* **22**(6), 655–667.

Gambardella, L. M., Taillard, É. and Agazzi, G. (1999), MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, *in* D. Corne, M. Dorigo and F. Glover, eds, 'New Ideas in Optimization', Kluwer Academic Publishers.

Gendreau, M., Hertz, A. and Laporte, G. (1992), 'New insertion and postoptimization procedures for the traveling salesman problem', *Operations Research* **40**(6), 1086–1094.

Gendreau, M., Hertz, A., Laporte, G. and Stan, M. (1998), 'A generalized insertion heuristic for the traveling salesman problem with time windows', *Operations Research* **43**(3), 330–335.

Glover, F. and Laguna, M. (1997), *Tabu search*, Kluwer Academic, Boston.

Golden, B. L. and Assad, A. A., eds (1988), *Vehicle routing: methods and studies*, North-Holland, Amsterdam.

Homberger, J. and Gehring, H. (1999), 'Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows', *INFOR* **37**(3), 297–318.

Larsen, J. (1999), 'Vehicle routing with time windows - Finding optimal solutions efficiently', *DORSnyt* **119**.

Lenstra, J. K. and Rinnooy Kan, A. H. G. (1981), 'Complexity of Vehicle Routing and Scheduling Problems', *Networks* **11**(2), 221–227.

Mullaseril, P. A., Dror, M. and Leung, J. (1997), 'Split-delivery routeing heuristics in livestock feed distribution', *Journal of the Operational Research Society* **48**(2), 107–116.

Rochat, Y. and Taillard, É. (1995), 'Probabilistic Diversification and Intensification in Local Search for Vehicle Routing', *Journal of Heuristics* **1**(1), 147–167.

Rousseau, L. M., Gendreau, M. and Pesant, G. (2002), 'Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows', *Journal of Heuristics* **8**(1), 43–58.

Shaw, P. (1998), Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, *in* M. Maher and J.-F. Puget, eds, 'Principles and Practice of Constraint Programming', Springer.

Solomon, M. M. (1987), 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations Research* **35**(2), 254–265.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F. and Potvin, J.-Y. (1997), 'A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows', *Transportation Science* **31**(2), 170–186.

Toth, P. and Vigo, D., eds (2002), *The Vehicle Routing Problem*, SIAM Society for Industrial and Applied Mathematics.