Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the VRPTW

Vitória Pureza, Reinaldo Morabito, Marc Reimann

1. Introduction

In vehicle routing problems, a set of routes is designed in order to meet customer demands. The literature in this field addresses situations with a variety of operational characteristics, such as the type of distribution (pickup and/or delivery), service nature (transportation of products/people or service delivery), constrained customer visiting hours, vehicle capacity limitations, maximum routing time, restricted types of vehicles for serving specific customers, among many others. The objective of a solution can be stated as that of minimizing fixed and variable total costs, such as the fleet size and the total distance of the tours. Most vehicle routing problems have a combinatorial nature and their inherent difficulty (the classic vehicle routing problem and most of its variants are NP-hard as pointed out in Lenstra and Rinnooy-Kan, 1981) and practical relevance explains the research efforts and continuous methodological development for their treatment.

This work addresses a variant of the vehicle routing problem with time windows (VRPTW) which allows a number of deliverymen (servicemen) to be assigned to each route in order to reduce service times. This variant is particularly relevant to situations with long service times (which are dependent on the quantity demanded) when compared with traveling times and it is derived from real life applications for which daily requests must be delivered on the same day, the total operation cannot be completed within the maximum routing time and violations to the latter are highly undesirable. Service times are thus also a function of the vehicle's crew size rather than fixed for a given request. The reduction of service times due to the assignment of extra deliverymen most often impacts the number of vehicles used and the total distance traveled. We call this variant the vehicle routing problem with time windows and multiple deliverymen (VRPTWMD).

This problem arises in situations faced by companies in Brazil and other developing countries that deliver goods in urban areas on a regular basis. Typical examples are soft drinks, juice drinks, mineral waters, beers and tobacco companies for which most customers consist of small and medium sized retailers (e.g., convenience stores, grocery stores, restaurants, snack bars and cafes, supermarkets, etc.) located in high-density populated regions. Given the difficulty in parking the vehicles in those areas due to scarcity of parking slots and heavy traffic, customers close to each other are seen as a demand cluster (supernode), and a single parking site is elected for the vehicle serving the cluster (note in this case that parking sites are the routed entities rather than the individual customers' locations). The goods are then delivered at each demand site of a given cluster by the vehicle's driver who visits the customers on foot, sometimes by making multiple tours from and to the parking site (Fig. 1). Thus, the delivery of goods is based on a two-phase approach: first the vehicles travel from the depot to the parking site assigned to the cluster, and then the vehicle's crew delivers the goods to the customer demand location on foot. In these situations, typical service times are relatively large when...
VRPTWMD as they consist of simultaneously locating facilities and designing delivery or collection routes and they do not involve the assignment of extra deliverymen to the vehicles in order to reduce service times. As far as we know, the VRPTWMD has not been reported in the literature and it has not been a feature included in any current commercial routing systems, despite its potential practical applications in the delivery or collection of goods in urban areas (Pureza and Morabito, 2008).

In this study we model the VRPTWMD with a homogeneous fleet as an extension of the two-index variable model presented in, e.g., Bard et al. (2002) and Cordeau et al. (2002) for the VRPTW. The modeling of the more general case of the VRPTWMD with a heterogeneous fleet is straightforward using a three-index variable formulation. We also propose two heuristic approaches based on tabu search and ant colony optimization, respectively. Using sets of instances with time windows based on the instances proposed in Solomon (1987) and some lower bounds for them, we compare the computational performance of the model and the heuristics. The remainder of this paper is organized as follows. Section 2 briefly discusses the underlying motivation for the VRPTWMD and presents the associated model with a homogeneous fleet. The proposed heuristic approaches are presented in Section 3. In Section 4, we discuss the computational experiments and results obtained from the chosen sets of instances. This numerical analysis is used to compare the solution methods with each other as well as to provide structural insights in the obtained solutions. Finally, in Section 5 we present our concluding remarks and discuss the next steps of this research.

2. Problem definition and modeling

The VRPTWMD can be more formally stated as the definition of minimum cost routes for a fleet of vehicles in order to satisfy a set of customer clusters with known demands for delivery (in the delivery version of the problem) or collection (in the pickup version) of goods. In each route, the service is delivered by the vehicle's crew whose size must not exceed the cabin's capacity. The pickup (or delivery) site corresponds to a single central depot, from which all vehicles depart and should return to. We assume that each cluster (consisting of one or various customers) is defined by the user and its total demand must respect the vehicle's capacity (expressed in terms of volume, weight, or number of pallets or containers) assigned to the route, as well as the total time of each route (comprising vehicle traveling times and service times in each cluster) should not violate a predefined limit associated to the end of the workday. Decisions such as selection of appropriate parking sites and allocation of customers to parking sites are taken before designing the vehicles routes. Therefore service times of each cluster are input parameters computed by the user, whose values depend on the crew's size, the deliverymen service strategy, and the cluster characteristics, such as the demand and the geographical dispersion of the customers.

All available vehicles may be used and each vehicle performs only one route. We also assume that the total demand of each cluster must be met by a single vehicle (i.e., non-split delivery or pickup) and within a given period of the day (time window constraint). Time windows are usually defined by the service deliverer and the cluster's customers. This restriction is particularly present in the pickup/delivery of goods in clusters made up of a large scale commercial establishment (such as supermarkets and hypermarkets) for which the cargo loading and unloading must occur before or after opening hours. It also appears in goods distribution in central areas of large cities, where the circulation of trucks is very often limited by law to light traffic hours.
Like most VRPT applications, the primary objective of the VRPTWMD is to minimize the fleet size (fixed cost). The second objective consists of the minimization of the total number of deliverymen, followed by the total distance traveled by the fleet (variable cost). Therefore, a solution with a small number of vehicles is always preferable to a solution requiring more vehicles, even if it demands a smaller number of deliverymen or total distance. More formally, the VRPTWMD consists of finding a solution $S$ that satisfies all restrictions and minimizes the function specified by the lexicographic order $f(S) =$ (number of vehicles, number of deliverymen, total distance). Alternatively, in cases where the cost parameters associated to the number of vehicles, number of deliverymen and distance traveled are well known, the objective function of the VRPTWMD can be appropriately defined as the minimization of the total cost, as presented below.

When the fleet is homogeneous, the problem can be modeled as an extension of the two-index variable model described in, e.g., Bard et al. (2002) and Cordeau et al. (2002) for the VRPTW. The network comprises $n$ nodes indexed by $i = 1, \ldots, n$; $i = 1$ represents the depot and $i = 2, \ldots, n$ refers to the parking sites (clusters). The following notation is used for the model’s description (either for the pickup version or delivery version of the problem):

**Parameters**
- $n$: number of nodes or clusters ($i = 1, \ldots, n$); the depot is represented by node $i = 1$, and the parking sites by nodes $i = 2, \ldots, n$.
- $L$: maximum crew size (i.e., the driver plus the extra deliverymen) that can be assigned to a single vehicle ($l = 1, \ldots, L$). If the crew size assigned to a vehicle is $l$, we say that this vehicle travels in mode $l$.
- $M$: maximum number of deliverymen to be assigned to the fleet.
- $c_i$: cost of a vehicle.
- $c_{2i}$: cost of a unitary distance traveled by the vehicles.
- $c_3$: cost of one deliveryman.
- $Q$: capacity of each vehicle.
- $T$: maximum duration of each route.
- $v$: average speed of the vehicles.
- $d_{ij}$: distance between nodes $i$ and $j$ ($i, j = 1, \ldots, n$).
- $d_{ij}$ may be different of $d_{ij}$ due to, e.g., road restrictions and one-way directions (asymmetric).
- $t_{ij}$: average direct travel time between nodes $i$ and $j$ ($i, j = 1, \ldots, n$).
- $t_{ij}$ may be different of $t_{ij}$ (e.g., $t_{ij} = t_{ij}$).
- $s_{ij}$: service time in mode $i = 1, \ldots, L$, deliverymen; it is assumed that $s_{11} = 0$.
- $b_{ij}$: non-negative demand (in the same units of capacity $Q$) of node $i = 1, \ldots, n$, with $b_{ij} \leq Q$ if $i = 2, \ldots, n$. $b_{ij}$ is equal to the quantities to be delivered (picked up) to the cluster associated to parking site $i$; it is assumed that $s_{11} = 0$.
- $a_{ij}$: earliest arrival time at node $i = 1, \ldots, n$; it is supposed that $a_{ij} = 0$.
- $a_{ij}$ is the quantity to be delivered in each cluster $i = 1, \ldots, L$ and leaves this node $i$ in the same mode $l$.

**Variables**
- $x_{ijl}$: number of deliverymen to node $i$ in mode $l$.
- $x_{ijl} = \begin{cases} 1, & \text{if the vehicle travels directly from node } i \text{ to node } j \text{ in mode } l \\ 0, & \text{otherwise} \end{cases}$.
- $i, j = 1, \ldots, n; i \neq j; l = 1, \ldots, L$.
- $t_{ij}$: service start time of the vehicle in mode $l = 1, \ldots, L$, at node $i = 1, \ldots, n$.
- $t_{ij}$ corresponds to the time the vehicle in mode $l = 1, \ldots, L$, returns to the depot.
- $y_{ijl}$: load in the vehicle in mode $l = 1, \ldots, L$, right after serving node $i = 1, \ldots, n$.

The VRPTWMD with homogeneous fleet is then formulated according to the following mixed 0-1 linear model:

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} \sum_{l=1}^{L} c_i x_{ijl} + c_{2i} \sum_{l=1}^{L} \sum_{j=1}^{n} d_{ij} x_{ijl} + c_3 \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{L} d_{ij} x_{ijl} + c_3 \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{L} x_{ijl} \\
\text{subject to:} & \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{L} x_{ijl} = 1, \quad j = 2, \ldots, n \\
& \quad \sum_{j=1}^{n} \sum_{i=1}^{n} \sum_{l=1}^{L} x_{ijl} = 1, \quad j = 2, \ldots, n \\
& \quad \sum_{j=1}^{n} \sum_{i=1}^{n} \sum_{l=1}^{L} x_{ijl} = 1, \quad i = 1, \ldots, n; \quad l = 1, \ldots, L \\
& \quad t_{ij} = t_{ij} \quad (t_{ij} = t_{ij} + t_{ij} x_{ij} - Q (1 - x_{ij}), \quad i = 2, \ldots, n; \quad j = 1, \ldots, n; \quad i \neq j; \quad l = 1, \ldots, L) \\
& \quad y_{ijl} = y_{ijl} + q_{ij} x_{ijl} - Q (1 - x_{ijl}), \quad i = 1, \ldots, n; \quad j = 2, \ldots, n; \quad i \neq j; \quad l = 1, \ldots, L \\
& \quad \sum_{j=1}^{n} \sum_{i=1}^{n} \sum_{l=1}^{L} x_{ijl} \leq M \\
& \quad x_{ijl} \in \{0, 1\}, \quad i = 1, \ldots, n; \quad i \neq j; \quad l = 1, \ldots, L \\
& \quad a_{ij} \leq t_{ij} \leq b_{ij} \quad (a_{ij} \leq t_{ij} \leq b_{ij}, \quad i = 1, \ldots, n; \quad i \neq j; \quad l = 1, \ldots, L) \\
\end{align*}
\]

where $N_0$ is a sufficiently large number (e.g., $N_0 = \max\{b_{ij} + (t_{ij} + t_{ij}) x_{ij} - Q (1 - x_{ij}), \quad i = 2, \ldots, n; \quad j = 1, \ldots, n; \quad i \neq j\}$). Note that if $N_0 = 0$, then constraint (5) is redundant even if $x_{ijl} = 1$. The objective function (1) minimizes the fixed and variable costs of the required fleet and the cost of assigned deliverymen. Constraint (2) ensures that only one vehicle in one mode enters in each cluster $j$ ($j = 1$), whereas constraint (3) ensures that only one vehicle in one mode leaves each cluster $i$ ($i = 1$). In other words, these two constraints impose that each node is served by exactly one vehicle in exactly one mode (it can be seen that one of these constraints is redundant for the model). Constraint (4) is the flow conservation equation that ensures the continuity of the route, that is, the same vehicle enters in node $i$ in mode $l$ and leaves this node $i$ in the same mode $l$.

Constraint (5) defines the relationship between the flow variables $x_{ijl}$ and the load variables $y_{ijl}$ and $a_{ijl}$, whereas constraint (6) ensures that the time window of the depot is met (given that $0 = a_{ij} \leq t_{ij} \leq b_{ij} = T$). Similarly, constraint (7) defines the relationship between the flow variables $x_{ijl}$ and the load variables $y_{ijl}$ and $a_{ijl}$, and it also eliminates the formation of subtours that do not contain the depot (as the load in the vehicle increases throughout a tour for the pickup version of the VRPTWMD). The delivery version of the problem, $q_{ijl}$ is the quantity to be delivered in each cluster $i$ and the load in the vehicle decreases throughout its delivery tour). Note that this constraint is defined for $l = 1$ (i.e., departure of the depot), but it does not need to be defined for $l = 1$ (return to the depot), given that $q_{ijl} = 0$. Constraint (7) ensures that the number of available deliverymen $M$ is not exceeded (note that this number can be less than $L$ times the number of used vehicles). Constraint (8) defines the domain of the variables, while constraint (9) ensures that the time windows of all nodes and the depot are satisfied, and the maximum capacity $Q$ of each vehicle is not exceeded. Observe that the time window constraints do not prevent a vehicle from arriving at node $i$ in a time instant less than $a_{ij}$, but they impede that the service in node $i$ starts at a time instant before $a_{ij}$. (In this case the deliverymen wait for time instant to initiate service in $i$).
3. Heuristic approaches

In this section we describe two heuristic approaches for solving model (1)–(9) for the VRPTWMD. The first one is based on tabu colony optimization and it is described in Section 3.1. The second one is based on ant colony optimization and it is presented in Section 3.2. Despite the specific methodology, both algorithms are based at their core of Solomon (1987)'s I1 insertion heuristic for solution construction and the $i$-interchange operators originally proposed by Osman (1993) with $z = 2$ for local solution improvement.

3.1. A TS-based heuristic approach

Starting with a constructive heuristic that generates routes with multiple deliverymen, a local search guided by an adaptive tabu search algorithm aims at solution improvement. The structure of the solution improvement procedure is general enough to be used either for reducing the number of routes, the number of deliverymen or the total distance. The general steps of the procedure are described in Fig. 2. The constructive heuristic, local search and adaptive tabu search procedures are presented in the following sections.

1. Generate a solution with a target fleet size and available deliverymen by applying the constructive heuristic (described in section 3.1.1), followed by the tabu search algorithm for number of routes reduction (described in section 3.1.2).
2. If there are unserved nodes, apply the constructive heuristic to include these nodes in the solution.
3. While there are unrouted nodes:
   3.1. Increase the fleet size by one unit and apply the constructive heuristic in order to include the unrouted nodes in the solution.
   3.2. Apply the tabu search algorithm for number of routes reduction.
4. Apply the tabu search algorithm for crew size reduction.
5. Apply the tabu search algorithm for distance reduction.

Even though the fleet size is considered unbounded in the model, we adopt a target size as means to keep the number of utilized vehicles small during the construction of the starting solution (step 1). This assumption, along with the greedy nature of the heuristic and the existence of time windows, does not ensure service for all requests, even with the maximum crew size on each route. Such drawback is actually the motivation for the procedure that decreases the number of vehicles in the current solution. Particularly, note that reducing the solution's number of routes addresses the two main objectives of VRPTWMD. It releases currently assigned vehicles for the insertion of unrouted nodes, and with each available vehicle, the number of assigned deliverymen is also reduced (step 2). In case there are still unserved nodes (step 3), the original fleet size is incremented, followed by the application of the constructive heuristic and the number of routes reduction algorithm in an alternate fashion. Although usually used for analyzing the impact of the fleet expansion, these steps often generate solutions in which all nodes are served with the target fleet size or less. The application of the tabu search algorithm for crew size reduction (step 4) and distance reduction (step 5) provides additional solution improvement.

3.1.1. Insertion based solution construction

The construction of a solution (steps 1.1, 2. and 3.1 in Fig. 2) is primarily performed by the algorithm described in Fig. 3. Its main feature is the generation of trying solutions in which the number of deliverymen in the partial solution $S$ is iteratively incremented (resulting in trying crew sizes) until all available deliverymen are assigned or the maximum crew size in each vehicle is reached or all nodes are routed in the solution. For a given iteration, the trying solution that produces the largest improvement according to the lexicographic ordering (number of unrouted nodes, number of routes, number of deliverymen, distance) is selected to replace $S$.

As aforementioned, we use Solomon’s I1 insertion heuristic for the VRPTW with a limited fleet to generate a starting solution with a single deliveryman on each route (step 1.1) or to insert unrouted nodes in the current solution (step 1.2). If all nodes are inserted, the route construction ends (step 2 followed by step 5). Otherwise, an iterative procedure that computes the impact of an additional deliveryman in the solution (step 4.3) is employed. An additional deliveryman is provisionally assigned to each route $j$ (one route at a time) if there are deliverymen available and the resultant $j$'s crew size ($\text{trycrew}_j$) does not exceed the maximum crew size ($\text{maxcrew}$) of the vehicle. In addition, since the increase in the crew size does not necessarily improve the solution, for each route $j$ we keep a record of the last crew size that failed to provide improvements ($\text{lastfail}_j$). If the current trying crew size for route $j$ fails
1. Let \( F \) be the target fleet size, \( N_u \) be the set of unruled nodes, \( t_{\text{crew}} \) be the number of available deliverymen (\( t_{\text{crew}} > F \)), \( t_{\text{crew}}_j \) be the number of deliverymen in a route \( j \), and \( max_{\text{crew}} \) be the maximum crew size.

1.1. If \( N_u = \emptyset \) apply the insertion heuristic to construct a starting solution \( S \) with one deliveryman in each route.

1.2. Otherwise, apply the insertion heuristic to include unruled nodes in \( S \).

1.3. Update \( N_u \) and \( t_{\text{crew}} \). Let \( r \) be the number of routes in \( S (r \leq F) \).

2. If \( N_u = \emptyset \), go to step 5.

3. Initialize \( last_{\text{fail}} \) (last crew size in route \( j \) that failed to improve the solution) = 0 and \( fail \) (number of routes that have already reached or tested \( max_{\text{crew}} \) = 0).

4. Repeat until \( fail = r \) or \( N_u = \emptyset \):

   4.1. For each route \( j \) in \( S \), make \( try_{\text{crew}}_j \) (number of deliverymen to be tested in route \( j \)) = \( crew_j \) and \( fail = 0 \).

   4.2. Initialize \( T^\ast \) (best tentative solution) = \( S \).

   4.3. For each route \( j \) in \( S \):

       4.3.1. If \( (\text{max}(try_{\text{crew}}_j + 1, last_{\text{fail}}_j + 1) \leq max_{\text{crew}} \) and \( t_{\text{crew}} \geq 1 \) generate a trying solution as follows:

       4.3.1.1. If \( \text{(last}_{\text{fail}}_j < try_{\text{crew}}_j \) (the last attempt of increasing the number of deliverymen in \( j \) improved the solution or this is the first attempt), make \( try_{\text{crew}}_j = \text{try}_{\text{crew}}_j + 1 \).

       4.3.1.2. Otherwise, if \( \text{(last}_{\text{fail}}_j > try_{\text{crew}}_j \) and \( \text{(last}_{\text{fail}}_j + 1 \leq max_{\text{crew}} \), make \( try_{\text{crew}}_j = last_{\text{fail}}_j + 1 \).

       4.3.1.3. Make \( try_{\text{crew}}_1 \) = \( try_{\text{crew}}_j \) (store trying crew size for route \( j \)).

       4.3.1.4. Reschedule route \( j \) with \( try_{\text{crew}} \) deliverymen in route \( j \).

       4.3.1.5. Apply the insertion heuristic to solution \( S \) to include unruled nodes and obtain a tentative solution \( T \).

       4.3.1.6. If \( T \) improves \( T^\ast \), make \( T^\ast \) (best tentative solution) = \( T \).

       4.3.1.7. For each route \( j \) in solution \( S \), make \( try_{\text{crew}}_j = crew_j \) (restore original crew sizes).

       4.3.2. Otherwise, make \( fail = fail + 1 \).

   4.4. If \( T^\ast = S \) (no improvement is obtained by adding a deliveryman to any route in solution \( S \)) make \( last_{\text{fail}}_j = try_{\text{crew}}_1 \), \( \forall j \) (the last crew size in route \( j \) that failed to improve the solution is the corresponding trying size).

   4.5. Otherwise, make \( S = T^\ast \), \( crew_j \) = number of deliverymen in route \( j \) of \( T^\ast \), and update \( N_u \) and \( t_{\text{crew}} \).

5. Return solution \( S \).

---

Fig. 3. Solution construction in TS approach.
Osman (1993) with $z = 2$. Specifically, the utilized moves are: (i) relocation of a single node $i$ in route $t$ to route $u$ (single insertion), (ii) relocation of two nodes $i$ and $j$ from route $t$ to route $u$ (double insertion), (iii) exchange of two nodes $i$ and $j$ in different routes $t$ and $u$, and (iv) a type (iii) move followed by a type (i) move (exchange/insertion).

Contrary to Bent and Van Hentenryck (2004), our approach does not allow infeasible solutions; however, a similar lexicographic ordering was used for move selection. It consists of the following three criteria: (i) maximum reduction in the number of vehicles, (ii) maximum difference of route sizes in the resulting solution, (iii) maximum reduction of the urgency of relocated nodes. The rationale of the second criterion is that the relocation of nodes from small routes to larger routes may eventually empty and eliminate some of the smaller routes. In the third criterion, the urgency of node $i$ in position $p$ of a route $k$ operating in mode $l$ is computed by $b_i - t^l_{ik}$, where $b_i$ is $i$'s latest arrival and $t^l_{ik}$ is the actual arrival of $i$ in $k$. The smaller the value of $b_i - t^l_{ik}$ is, the greater the urgency of node $i$.

The second and third criteria give thus higher priority to the relocation of urgent nodes from smaller routes to positions in larger routes that reduce their original urgency.

For the reduction of the crew size (step 4 in Fig. 2), the four types of moves comprise an additional step in which the number of deliverymen of each affected route is iteratively decremented until a further reduction results in an infeasible route. The lexicographic ordering described above is then modified by including the resulting crew sizes as a fourth criterion. For distance reduction (step 5 in Fig. 2), moves are selected according to the lexicographic order criteria: (i) number of vehicles, (ii) number of deliverymen, (iii) total distance.

### 3.1.3. The adaptive tabu search procedure

We use an adaptive tabu search approach proposed in França et al. (1999) for the capacitated clustering problem in order to guide the local search process. The approach is characterized by an integrated intensification/diversification mechanism that changes tabu search parameters (Glover and Laguna, 1997) based on the analysis of search trajectory patterns (Charleston, 1995). Basically, it is assumed that general search trajectory patterns (the curve of solution cost vs. iteration) reflect the restrictiveness level imposed by the tabu framework design or parameters values, e.g. by tabu tenure and tabu activation rules. In order to identify the current trajectory pattern, the search process is dynamically divided into stages, and for each two consecutive stages $g - 1$ and $g$, the current average solution cost in stage $g$ is compared to the previous average solution cost in stage $g - 1$. If these averages values are approximately the same, the search describes a stagnated trajectory. On the other hand, if the current average is larger than the previous average, we say that an ascent trajectory is taking place.

Similarly, a descent trajectory is identified by a current average smaller than the previous average.

Once the trajectory pattern is identified, changes in restrictiveness levels are prescribed not only in terms of nature (increasing or decreasing levels), but also in degree and duration. The period in which changes are applied is dynamic and defined by the expression $\sigma_{r\min}$, where $\sigma_{r\min}$ is a tuning factor that depends on the prescribed change of restrictiveness resulting from the observation of trajectory pattern $m$. It should be noted that prior to a trajectory evaluation, it is verified if the last stage corresponds to an improvement phase; in this case, levels of restrictiveness and tuning factor value are also set.

As one can see, the approach reacts to improvement phases, stagnation, ascent trajectories and descent trajectories, which means that four possible tuning factor values ($m = 1,2,3,4$) are used along the algorithm in order to define the duration of restrictiveness levels changes. The period of application of a new operational setting corresponds to a new stage ($g + 1$) and, once the end of this stage is reached, a new trajectory pattern evaluation is performed using the average costs of stages $g$ and $g + 1$. A description of the algorithm is presented in Fig. 4.

In our approach added (removed) arcs in recent moves are labeled as tabu-active and restrictiveness levels are controlled by the tabu activation rule. It prescribes the tolerable number of tabu-active arcs for each of the four move types in a given search stage. For a move involving $\phi$ arcs, possible tolerance values vary from 0 to $\phi$; for single insertion, exchange, double insertion, and exchange/insertion move types, $\phi$ is equal to 6, 8, 12 and 14, respectively. Note that the smaller the tolerance value, the more restrictive the search process is. Tolerance values for single insertion, exchange, double insertion, and exchange/insertion moves in a given search stage are provided by parameters $TI$, $TE$, $TDI$, and $TEI$, respectively. In step 2, standard tolerance values are set to $(TI,TE,TDI,TEI) = (3,4,6,7)$, corresponding to half the maximum values these tolerances can have. These values are applied along the first two search stages. The first stage comprises all solutions between the starting solution and the first local optimum, while the second stage starts with the solution that follows the first local optimum and ends with the solution found $h$ iterations ahead (that is, the tuning factor is 1).

In each iteration, the number of nodes of the smaller route is used as the solution cost for the trajectory patterns analysis. In step 4.1, if an improvement phase or a descent trajectory are identified, tolerance values correspond to $(TI,TE,TDI,TEI) = (6,7,6,7)$ and $(TI,TE,TDI,TEI) = (4,5,4,5)$, respectively. The tuning factors in these cases $(\sigma_{r1}, \sigma_{r2})$ are set to 2 and 1, respectively. Maximum restrictiveness levels $((TI,TE,TDI,TEI) = (0,0,0,0))$ with $\sigma_{r1} = 0.5$ are imposed when search stagnation takes place, specifically, when

1. Read the input data.
2. Initialize current iteration and set standard parameters.
3. From the starting solution provided by the constructive heuristic (discussed in section 3.1.1), proceed with the search for two stages.
4. Repeat until no feasible moves are available or the maximum runtime is elapsed:
   4.1 If solution improvement is verified in the last stage, update the best found solution. Otherwise, identify current pattern described by the last two stages.
   4.2 Set parameters according to the trajectory pattern or to an improvement phase. Apply the setting for the prescribed period of application, obtaining a new stage.
5. Return the best found solution.

Fig. 4. A high-level description of the tabu search algorithm.
the absolute value of the percent deviation of the solution’s current average from the previous average (apd) is less than or equal to a given threshold ST. That is, stagnation is identified when:

\[ |\text{apd}| = 100 \times \frac{\text{current average} - \text{previous average}}{\text{previous average}} \leq ST \]

In the occurrence of ascent trajectories, the prescribed (Ti, TE, TDI, TEI) values depend on the range of apd values. The general idea is that the larger the apd is, the largest is the increase in restrictiveness levels between the two last search stages, and more vigorous should be the relaxation imposed, and in this case, the tuning factor (σ3) is set to 1. Given the many possibilities for setting the parameter values, we performed a number of preliminary experiments with test instances in order to define ST, tolerance values, and deviation ranges written as a function of ST. A suitable setting (with ST = 2%) is given in Table 1. The lower limit of the first apd range (6ST = 2%) is the largest difference found between averages, characterizing what we call a “highly” ascent trajectory. We discretize the following apd ranges in a rather simple fashion until the lower limit of the last range is equal to the stagnation threshold ST. Following the original approach, single insertion moves have their tolerance value slowly decreased from one apd range to the other, starting with the maximum tolerance value 6. Tolerance 3 was set for the last apd range since smaller values usually produce increasing solution costs. For the other three move types, experiments have shown that relatively large tolerance values in the first range frequently led to cycling in the next search stage, especially when the move’s number of involved edges is large. Cycling was eliminated when we applied tolerance values slightly smaller than the set of yet unserved nodes Ni, a node i is inserted probabilistically into the current route k according to the following random proportional rule:

\[ P_{ik} = \frac{\kappa_{ik}}{\sum_{h \in N_{i}} \kappa_{ih}} \]

where \( \kappa_{ik} \) is a function of both the heuristic information and the pheromone level associated with inserting node i into route k. The computation of \( \kappa_{ik} \) differs depending on whether a new route is initialized or a current route is extended. In the former case it is given by:

\[ \kappa_{ik} = \frac{1}{d_{ii}} (\tau_{ii} + \tau_{i0}) \]

for \( i \in N_{u} \)

while in case of a route extension it is computed as:

\[ \kappa_{ik} = \max_{j:R_{i} \ni j} \left\{ \left[ \frac{\tau_{ij} + \tau_{ij}}{2\tau_{ij}} \right] \right\} \]

for \( i \in N_{u} \)

where \( R_{i} \) denotes the set of customers already assigned to route k after which node i could be feasibly inserted, and \( j' \) denotes the current successor of j in route k, while the pheromone information \( \tau_{ij} \)

3.2. An ant colony optimization approach

Our second heuristic approach for the VRPTWMD consists of an implementation of ant colony optimization (ACO). ACO was first proposed in Dorigo et al. (1996) as a population-based metaheuristic. Its motivation stems from the underlying metaphor concerning the collective behavior of real ant colonies leading to the exploitation of rich food sources. More precisely, through a trail laying/trail following mechanism promising (shortest) paths from the nest to a nearby food source are reinforced. An overview of different variants of ACO can be found in Dorigo and Stützle (2004). For some basic versions asymptotic convergence results are provided in Gutjahr (2000), Gutjahr (2002) and Stützle and Dorigo (2002).

The ACO approach used in this research is a modification and adaptation of the algorithm originally proposed in Reimann et al. (2002) for the VRPTW. In Reimann et al. (2003) it was applied to several versions of vehicle routing problems showing its versatility, and in Reimann and Ulrich (2006) it was used to analyze different backhauling strategies in the VRPTW. For the purpose of this research we have modified the objective function and the local search to allow the minimization of the number of deliverymen as well. A high-level description of the algorithm is shown in Fig. 5 followed by the discussion of the main components of the algorithm.

3.2.1. Insertion based solution construction

As in our tabu search approach solution construction is based on the 11-heuristic for the VRPTW from Solomon (1987). In this algorithm routes are constructed sequentially one by one. In the context of ACO this originally deterministic algorithm was modified in the following way (see also Reimann et al. (2002)). From the set of yet unserved nodes Ni, a node i is inserted probabilistically into the current route k according to the following random proportional rule:

\[ P_{ik} = \frac{\kappa_{ik}}{\sum_{h \in N_{i}} \kappa_{ih}} \]

1. Read the input data.
2. Initialize parameters and pheromone matrix (discussed in section 3.2.3).
3. Repeat until a pre-specified stopping criterion is met:
   3.1. For each ant:
      3.1.1. Construct a feasible solution (discussed in section 3.2.1.)
      3.1.2. Apply local search (discussed in section 3.2.2.).
      3.1.3. Update the best found solution (if applicable).
   3.2. Update the pheromone matrix (discussed in section 3.2.3).
4. Return the best found solution.

![Fig. 5. A high-level description of the ACO algorithm.](image-url)
represents the learned desirability of visiting a node \( i \) immediately after a node \( j \) on the same route. Thus, for each unserved node \( i \), the best insertion position along route \( k \) is selected deterministically and the random proportional rule described above is only used to select which node to insert. Only if no more feasible insertions are possible in a route a new route is initialized. The algorithm stops once all nodes have been assigned to routes.

Differently from the strategy employed in the tabu search approach, the solution construction assigns the maximum crew size (if available) to each route in order to keep the required fleet size as small as possible. Still, the solution returned by an ant may feature routes that actually can be feasibly performed with a smaller number of deliverymen. As in the tabu search approach, in such a case the number of deliverymen is iteratively decremented until a further reduction results in an infeasible route. The complete solution construction procedure is depicted in Fig. 6.

3.2.2. Local search

After an ant has constructed a feasible solution, this solution undergoes local search to improve its quality. For this end, we sequentially apply the \( k \)-interchange operators (Osman, 1993) with \( k = 1, 2 \). Infeasible solutions are forbidden and a first improvement strategy is employed. Given the lexicographic ordering of the three objectives, we first try to move \( k \leq 2 \) nodes from the routes with the smallest number of nodes to the other routes with the goal to eliminate some of the routes. Then we apply move and swap operators with \( k = 1 \) to improve the routing.

For this study we modified this local search by adding a move neighborhood with \( k = 1 \) that aims at reducing the number of deliverymen. Precisely, we eject one customer from a route with 2 or 3 deliverymen if this reduces the deliverymen on this route. Then we try to insert this node into another route. If such an insertion is possible without (or with a smaller) increasing the number of deliverymen on the new route, the move is accepted.

3.2.3. Pheromone initialization and update

In the constructive phase of the ACO algorithm, decisions are based on both heuristic information and the pheromone values as described above. At the end of each iteration, that is, once all ants have gone through solution construction and local search, the pheromone update procedure is applied to these pheromone values. The pheromone management used in our algorithm is related to the Hypercube Framework presented in Blum and Dorigo (2004) and the Max–Min Ant System (see, e.g., Stützle and Hoos, 1999) and a variant of it was first presented in Reimann (2003). Formally, the pheromone update rule can be written as:

\[
\tau_{ij} = \rho \tau_{ij} + (1 - \rho) \Delta \tau_{ij}, \text{ for } (i, j) \in E
\]

where \( E \) is the set of all edges, \( 0 \leq \rho \leq 1 \) is called the trail persistence and \( \Delta \tau_{ij} \) is the amount of reinforcement, which is defined as:

\[
\Delta \tau_{ij} = \begin{cases} 
1 & \text{if } (i, j) \in S^* \\
0 & \text{otherwise}
\end{cases}
\]

where \( S^* \) is the best solution found up to the current iteration (regardless if it was found in the current iteration or earlier). At the beginning of the run, all pheromone values are initialized as:

\[
\tau_{ij} = 1, \text{ for } (i, j) \in E
\]

4. Computational experiments

In this section, we compare the computational results obtained by solving model (1)–(9) of Section 2 with an exact branch-and-cut algorithm (B&C) and with the heuristic approaches of Section 3 (algorithms TS and ACO). For these experiments, we used the following VRPTW data sets proposed in Solomon (1987): (i) R1 and R2 (randomly generated geographical distribution of customer nodes), (ii) C1 and C2 (clustered problem sets) and (iii) RC1 and RC2 (mix of random and clustered structures), with \( n = 100 \) nodes and made up of 12, 11, 9, 8, 8 and 8 instances, respectively. The maximum crew size in each vehicle was considered equal to \( L = 3 \). The original data was maintained except for the service times in each node \( i \), which were replaced by:

\[
ts_i = \min\{r_s + q_i, T - \max\{a, tv_{i1} - tv_{i1}\}\}
\]

Fig. 6. Solution construction and reduction of deliverymen by each ant in the ACO algorithm.
where \( q_i \), as before, is the product demand of node \( i \) and \( r_s \) is the rate of service (i.e., number of products picked up per time unit in each node – in the experiments we used \( r_s = 2 \)). We remember that \( T \) is the maximum return time for all vehicles to the depot (as in Section 2, we considered \( T = b_d \)), \( a_i \) is the earliest arrival time of the time window of node \( i \), and \( t_{ri} \) is the travel time between the depot and node \( i \) (i.e., \( t_{ri} = \frac{q_i}{r_s} \); in the experiments we used \( v = 1 \)). Note that in (13) is proportional to the demand \( q_i \) of node \( i \) when the first term on the right-hand-side of (13) is the minimum. The second term in (13) is to ensure that this service time does not turn the instance infeasible regarding the maximum return time \( T \). In the experiments, the value of \( t_{ri} \) in (13) was considered the service time in node \( i \) and in mode 1, i.e., \( t_{ri} = t_{si} \). For the sake of simplicity, for the cases with two and three deliverymen (modes 2 and 3), we simply divided \( t_{ri} \) by 2 and 3, so that \( t_{ri} = \frac{n}{i} \) for \( i = 1, 2, 3 \). We note that in practice \( t_{ri} \) may not be linearly dependent on \( i \).

The modeling language GAMS with the solver CPLEX 11.0 (Brooke et al., 1992) was used to implement and solve the VRPTW model. We arbitrarily chose a set of cost values \((c_1 = 1, c_2 = 0.0001 \) and \( c_3 = 0.1) \) so that the model solution \( S \) minimizes the function specified by the lexicographic order \( f(S) = (\text{fleet size}, \text{number of deliverymen}, \text{total distance}) \), in accordance with the objective of the heuristics. The CPLEX branch-and-cut ran with all its default parameters, but with the option of parallel processing using 4 threads and within a runtime limit of 36,000 seconds (10 hours). The experiments were performed in a computer Intel Core2 2.67 GHz with 12 GB RAM.

Algorithm TS was coded in Borland Delphi 7 (Pascal) using as the target fleet size the number of vehicles required in the best solution reported for the VRPTW (http://www.sintef.no/Project-web/TOPProblems/VRPTW/Solomon-benchmark/). In the experiments, each instance was solved using up to five different initial solutions, obtained with five different sets of parameters \([\sigma, \delta, \alpha, \omega] \) values utilized in Solomon’s 11 heuristic. Tabu parameters \( h_{rz} \) and the tabu tenure are drawn from range \([h_{min}, h_{max}] = \left[\sqrt{\pi}, 2\sqrt{\pi}\right] \) and \( \left[\frac{2}{\pi}, \frac{3.2}{\pi}\right] \), respectively. For each individual application of the algorithm of reduction of the number of vehicles, crew or distance, we set a runtime limit of 60 seconds. The ACO was coded in C and run with a time limit of 600 seconds under the following (standard) parameter settings: 50 ants and a trail persistence of \( \rho = 0.975 \). The experiments were run on a microcomputer Intel Core2 2.4 GHz with 2 GB RAM, and the performances of the approaches were evaluated in terms of each of the objectives (fleet size, number of deliverymen, total distance) and the computer runtime requirements.

First we analyze the results obtained for the instances of set R1, which are characterized by tight time windows and randomly generated geographical distribution of the nodes. Short planning horizons (set type 1) better reflect the importance of the service times since the routes are shorter in general. Moreover, since the nodes are parking sites from which clusters of customers are visited, it seems more straightforward that these parking sites are randomly distributed (set type R) rather than clustered. Table 2 presents the results of B&C, TS and ACO for this set considering the number of deliverymen sufficiently large (i.e., \( M = 50 \)). For each instance and method, columns Veh, Dmen, Dist and Cpu depict the best solution regarding the number of vehicles used, the number of deliverymen assigned, the total distance traveled and the runtime required (in seconds), respectively. The solutions of B&C, TS and ACO in Table 2 are also compared to the solutions obtained considering only one deliveryman in each route (algorithm TS1), which corresponds to the result of step 1.1 of the constructive heuristic in TS (Fig. 3). The best solution for each instance is highlighted in bold. For heuristic TS1, the number of deliverymen is replaced by the number of unserved clusters within the maximum route time (column Unsv).

The application of heuristic TS or ACO increases by approximately 43.5% the number of nodes served compared to the solutions of TS1 (each vehicle manned by one deliveryman; see last row of Table 2). This improvement is obtained at the expense of a relatively small increase (less than 5%) of the number of vehicles used, but a substantial increment of more than 160% of the number of deliverymen assigned to these vehicles and more than 10% in the total distance traveled. It is interesting to note that in 7 out of the 12 instances, the total number of vehicles used coincides with the original fleet size. The mean runtime of algorithm TS1 is less than 1 second, while algorithm TS and ACO require approximately 10 minutes on average per starting solution.

Regarding B&C, the software GAMS/Cplex was able to solve optimally only instance R101 within the time limit of 10 hours. Note that for instance R101, algorithms TS and ACO found solutions using the optimal number of vehicles and deliverymen, and an increase of 1% and 0.1% in distance traveled, respectively. For the remaining instances, the best feasible solutions found by B&C are worse than the ones produced by heuristics TS and ACO, with differences in the number of vehicles varying from 2 to 6 vehicles. Both TS and ACO had equivalent performances in terms of the number of vehicles (TS outperformed ACO in only 2 out of the 12 instances), number of deliverymen (ACO was slightly better than TS in 7 out of 12 instances) and distance (TS was slightly better than ACO in 7 out of 12 instances). Note also that the average results of these 12 instances obtained by TS and ACO and presented in the last row of Table 2 are close to each other.

The best lower bounds produced by B&C after 10 hours of computing time are far from the best feasible solutions, with relative

![Table 2](image)
The average results of B&C, TS and ACO for data sets R1, R2, C1, C2, RC1 and RC2, together with the average of the best results for set R1. Note that the average values of TS are slightly better than the ones of ACO, although one algorithm does not dominate the other. The solutions of ACO outperformed the solutions of TS in 8 out of the 12 instances of set R1 and in 8 out of the 11 instances of set R2. Regarding sets C1 and C2, the TS solutions outperformed the ACO solutions in 5 out of the 9 instances of C1 and in 6 out of the 8 instances of C2. For sets RC1 and RC2, ACO was better than TS in 5 out of the 8 instances of RC1, but worse than TS in 5 out of the 8 instances of RC2.

Table 4 presents the average values of the lower and upper bounds for the solution values of sets R1, R2, C1, C2, RC1 and RC2 obtained by B&C. Note that as in Table 3 the gaps between these bounds are relatively large. Best z corresponds to the average of the best solution values (z values) obtained by TS and ACO, while the remaining columns depict the average values of the numbers of vehicles and deliverymen, the distances traveled and the execution times, considering the five runs of the algorithms. The best average solution for each instance is highlighted in bold. Note that the standard deviations are relatively small if compared to the averages. Note also in the last row of Table 3 that ACO presents a slightly better average performance than TS.

Due to the difficulties of GAMS/Cplex to solve these instances, we performed other experiments with set R1 considering only the first 25 and 50 nodes (out of the 100 original nodes) of each instance. Only 3 (R101_25, R105_25 and R101_50) out of the 24 examples with 25 and 50 nodes were optimally solved with GAMS/Cplex within the time limit of 10 hours, which reveals the difficulties of the problem solving with the branch-and-cut method, even for problems of relatively small number of nodes. For two of these three examples, algorithms TS and ACO were able to find solutions with the optimal number of vehicles and deliverymen in less than 1 minute, but with increases of less than 2% and 0.7% in the total traveled distances, respectively. For the remaining examples, algorithms TS and ACO typically produced much better solutions than the best feasible ones found by B&C (in some cases with reductions up to 2 vehicles) consuming 88 and 26 seconds on average, respectively. Similarly to the results of Table 3, the best lower bounds produced by B&C for these instances with 25 and 50 nodes are also far from the best feasible solutions, with gaps typically higher than 30%.

Table 4 presents the average of the best results of B&C, TS and ACO for the instances of the remaining sets R2, C1, C2, RC1 and RC2, together with the average of the best results for set R1. Note that the average values of TS are slightly better than the ones of ACO, although one algorithm does not dominate the other. The solutions of ACO outperformed the solutions of TS in 8 out of the 12 instances of set R1 and in 8 out of the 11 instances of set R2. Regarding sets C1 and C2, the TS solutions outperformed the ACO solutions in 5 out of the 9 instances of C1 and in 6 out of the 8 instances of C2. For sets RC1 and RC2, ACO was better than TS in 5 out of the 8 instances of RC1, but worse than TS in 5 out of the 8 instances of RC2.

Table 5 presents the average values of the lower and upper bounds for the solution values of sets R1, R2, C1, C2, RC1 and RC2. Note that as in Table 3 the gaps between these bounds are relatively large. Best z corresponds to the average of the best solution values (z values) obtained by TS and ACO, while the remaining columns depict the average values of the numbers of vehicles and deliverymen, the distances traveled and the execution times, considering the five runs of the algorithms. The best average solutions (of the five solutions) are highlighted in bold.

TS and ACO were also applied to sets R1, R2, C1, C2, RC1 and RC2 with their original service times and with a single deliveryman in order to evaluate the performance of the algorithms when tackling the classical VRPTW. The average number of vehicles and the average distance traveled obtained by TS and ACO for these sets were 7.41 and 1040.7, and 7.59 and 1016.8, respectively, while the best known results are 7.23 and 1021.1, respectively (http://www.sintef.no/Projectweb/TOP/Problems/VRPTW/Solomon-benchmark/). Since the best-known solutions are taken from several papers featuring highly specialized algorithms for the VRPTW and our approaches were designed for tackling the additional
decision of assigning deliverymen, these results can be considered reasonable.

4.1. Trade-off between the number of vehicles and the number of deliverymen

In logistic systems in which labor costs are high and/or the workforce is scarce but not clearly limited, it may be valuable to present solutions that minimize the number of vehicles for different total numbers of available deliverymen. Or, alternatively, solutions that minimize the total number of deliverymen for different numbers of vehicles may also be useful. Even though the calculation of such efficient solutions is usually a time consuming process, it clearly reveals the trade-off between the minimum number of vehicles and the minimum number of deliverymen, allowing the decision maker to choose his/her most preferred efficient solution in the whole set.

For this end, we used model (1)–(9) to solve instance R101 with different values of the total number of available deliverymen. Starting \( M = 50 \) (a sufficiently large number for this problem instance) and decreasing the value of \( M \), one by one, until \( M = 34 \) (which makes the problem instance infeasible), the CPLEX branch-and-cut was able to find all optimal solutions within a time limit of one hour. Fig. 7 presents the results obtained – note that the solutions with \( M = 46 \) to 50 are clearly dominated by the solution with \( M = 45 \) (similarly for the other dominated solutions with \( M = 39, 41, 43 \) and 44). Data series Veh, Dmen, Dist and Cpu indicate for each \( M \), the best values obtained for the number of vehicles used \( \left( \sum_{j=1}^{n} \sum_{l=1}^{L} x_{jil} \right) \), the number of deliverymen assigned \( \left( \sum_{j=1}^{n} \sum_{l=1}^{L} x_{jil} \right) \), and the total distance traveled \( \left( \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{L} d_{ij} x_{jil} \right) \), respectively. Note that as the value of \( M \) is decreased, the figure depicts how much the optimal values of the number of vehicles and total distance need to be increased in order to compensate the reduction in the optimal value of the number of deliverymen. Except for the solution obtained with \( M = 35 \), where there are no vehicles with three deliverymen, the solutions for all other \( M \) values combine routes with one, two and three deliverymen.

5. Concluding remarks

In this study we introduced the VRPTWMD and formulated it as an extension of a two-index variable model for the VRPTW. We also proposed two solution approaches tailored for the problem based on tabu search and ant colony optimization. The impact of the use of extra deliverymen in route planning was assessed and discussed by means of computational experiments comprising 100 demand cluster instances derived from classic VRPTW. The results indicated the benefits of the two heuristics. Comparison with lower and upper bounds for the optimal number of vehicles obtained from the model revealed that the two heuristic approaches are capable of producing good solutions for the examined instances. The performances of the heuristics were comparable when considering the lexicographic objective function (number of vehicles, number of deliverymen and total distance) and one algorithm does not dominate the other.

An interesting perspective for future research is to extend the model and the heuristic approaches to deal with more general VRPTWMD cases of heterogeneous fleet and multiple depots. Other interesting lines of research are to adapt the approaches to deal with the problem in situations with simultaneous (or mixed) pickup and delivery routes and when the objective is to maximize the total number of serviced customers within regular working hours. Further work to support decisions of the definition and sizing of the clusters, as well as to study in practice how the service time of the cluster varies as the number of deliverymen is increased and evaluate the application of these approaches for solving the VRPTWMD in practice, is also on our research agenda.

| Table 5 | Additional results of B&C, TS and ACO for data sets R1, R2, C1, C2, RC1 and RC2. |

<table>
<thead>
<tr>
<th>Set</th>
<th>B&amp;C</th>
<th>TS best and average values</th>
<th>ACO best and average values</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>4.37</td>
<td>20.42</td>
<td>0.81</td>
</tr>
<tr>
<td>R2</td>
<td>0.17</td>
<td>10.61</td>
<td>0.98</td>
</tr>
<tr>
<td>C1</td>
<td>0.82</td>
<td>11.90</td>
<td>0.93</td>
</tr>
<tr>
<td>C2</td>
<td>0.08</td>
<td>11.65</td>
<td>0.99</td>
</tr>
<tr>
<td>RC1</td>
<td>3.08</td>
<td>19.61</td>
<td>0.88</td>
</tr>
<tr>
<td>RC2</td>
<td>0.08</td>
<td>13.02</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Fig. 7. Computational results of instance R101 for different values of \( M \).
Acknowledgments

The authors thank the two anonymous referees for their useful comments and suggestions, and Sebrae (Projeto Transmute) and Bebidas Ipiranga for their collaboration. This study was partially supported by CNPq and FAPESP.

References