An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen

Aldair Álvarez, Pedro Munari

Industrial Engineering Department, Federal University of São Carlos,
Rodovia Washington Luis - Km 235, CEP: 13565-905, São Carlos-SP, Brazil
aldair@dep.ufscar.br, munari@dep.ufscar.br

Abstract

The vehicle routing problem with time windows and multiple deliverymen (VRPTWMD) is a variant of the vehicle routing problem with time windows in which service times at customers depend on the number of deliverymen assigned to the route that serves them. Hence, in addition to the usual routing and scheduling decisions, the crew size for each route is also an endogenous decision. This problem is commonly faced by companies that deliver goods to customers located in busy urban areas, a situation that requires nearby customers to be grouped in advance so that the deliverymen can serve all customers in a group during one vehicle stop. Consequently, service times can be relatively long compared to travel times, which complicates serving all scheduled customers during regular work hours. In this paper, we propose a hybrid method for the VRPTWMD, combining a branch-price-and-cut (BPC) algorithm with two metaheuristic approaches. We present a wide variety of computational results showing that the proposed hybrid approach outperforms the BPC algorithm used as standalone method in terms of both solution quality and running time. These results indicate the advantages of using specific algorithms to generate good feasible solutions within an exact method.

Keywords: vehicle routing; multiple deliverymen; hybrid method; branch-price-and-cut; metaheuristics; column generation.

1 Introduction

Vehicle routing problems have been widely studied over the last decades due to practical relevance as well as their challenging formulations (Irnich et al., 2014). In this paper, we address the Vehicle Routing Problem With Time Windows and Multiple Deliverymen (VRPTWMD), an extension of the Vehicle Routing Problem with Time Windows (VRPTW) recently introduced in the literature (Pureza et al., 2012). In the VRPTWMD, the service time at the operational points (e.g., customers) of a route depends on the crew size assigned to that route. Therefore, in addition to the typical scheduling and routing decisions, the VRPTWMD treats the number of deliverymen assigned to each route as a decision variable.

The VRPTWMD is motivated by the real-life delivery and/or pick-up of goods in congested urban areas where promptly available parking spaces near customers’ locations can be very limited. To deal with this problem, nearby customers are considered as a single cluster and a common parking location is chosen for the customers in the cluster. Then, the vehicle parks at this common location and the workers deliver (and/or collect) the products on foot to all customers in the
cluster. Because of this operational mode, service times at the clusters can be very long, usually corresponding to a large percentage of the total route duration. This can compromise the service of all customers during regular working hours, especially when only one deliveryman is assigned to the routes. In this context, using multiple deliverymen becomes an important feature to reduce the service time at each customer cluster.

In this paper, we propose a hybrid method based on a cooperative scheme between a branch-price-and-cut (BPC) algorithm and two metaheuristic approaches for the VRPTWMD. The metaheuristic approaches are based on Iterated Local Search (ILS) and Large Neighborhood Search (LNS) respectively, and use tailored operators within the search. The BPC algorithm uses state-of-the-art features such as an interior-point stabilization strategy in the column generation process, separation of valid inequalities using well-centered points of the feasible sets, strong branching and a MIP-based primal heuristic. In the hybrid method, metaheuristics can improve the capacity of the BPC to generate good feasible solutions at an earlier stage while retaining the advantage of the BPC in providing good lower bounds.

There are few papers proposing solution methods for the VRPTWMD. Pureza et al. (2012) model this problem based on the standard arc flow formulation for the VRPTW (Desaulniers et al., 2014). As current general-purpose integer programming solvers are not effective for solve such formulations within a reasonable running time, the authors propose two metaheuristic approaches to solve the problem. The first one is a Tabu Search (TS) algorithm characterized by an adaptive mechanism that changes the parameters of the search based on an analysis of the search trajectory pattern. The second solution approach is an Ant Colony Optimization (ACO) algorithm that constructs solutions through a probabilistic insertion mechanism. Senarclens and Reimann (2014) propose two metaheuristic algorithms based on ACO and GRASP to solve the VRPTWMD. To make the algorithms comparable, the metaheuristics rely on the same constructive heuristic and local search components. Alvarez and Munari (2016) propose two metaheuristic approaches based on ILS and LNS for the VRPTWMD. In addition to the typical components of these metaheuristics, both approaches use a set of specific heuristics designed to enhance its performance, namely, route reduction heuristic and deliverymen reduction heuristic. The latter approaches are the current state-of-the-art metaheuristic algorithms for the VRPTWMD. Ferreira and Pureza (2012) develop two heuristic algorithms for a variant of the problem without time windows and drop the requirement of visiting all customers. In their study, the first algorithm is an extension of the savings algorithm (Clarke and Wright, 1964) and the second is a TS heuristic that uses an adaptive mechanism as in Pureza et al. (2012). Munari and Morabito (2016) propose a BPC method for the VRPTWMD, which is the first exact algorithm proposed specifically for this problem. The method relies on advanced components such as an interior point stabilization strategy in the column generation method, subset row cuts as valid inequalities and different types of branching rules. As reported by the authors, the method was able to find optimal solutions for the first time for several instances. Moreover, improved upper bounds were obtained by the method.

Although the VRPTWMD can be used as basis for practical and theoretical applications, to the best of our knowledge, no hybrid methods have been proposed in the literature to solve it. Hence, the main contribution of this paper is that we develop a hybrid method to solve it, reporting computational results obtained on instances from the literature. In addition, we provide experimental results to assess the benefits of using extra deliverymen in scenarios allowing different limits on the crew size. In the past several years, hybrid methods combining exact and metaheuris-
tics approaches have become popular for solving vehicle routing problems (Kramer et al., 2015; Cacchiani et al., 2014; Nishi and Izuno, 2014; Villegas et al., 2013; Subramanian et al., 2012; Salari et al., 2010; Alvarenga et al., 2007; Danna and Le Pape, 2005). These hybrid methods attempt to simultaneously exploit the advantages of their components to obtain better solutions than those obtained by those same components as standalone approaches.

The remainder of this paper is organized as follows. In Section 2, we describe the VRPTWMD and a set partitioning formulation for the problem. The hybrid method introduced to solve the VRPTWMD and its components are detailed in Section 3. The results of our computational experiments are described in Section 4. Finally, in Section 5, we present the main conclusions of this research.

2 Problem description

In the VRPTWMD, an optimal solution must satisfy the constraints of vehicle capacities, time windows and available deliverymen while minimizing the total cost composed by the number of vehicles, number of deliverymen and traveled distance costs of the routes. Similar to previous studies on this problem (Pureza et al., 2012; Senarcens and Reimann, 2014; Álvarez and Munari, 2016; Munari and Morabito, 2016), we assume that the clusters of customers are defined in advance, thus, decisions concerning the selection of parking locations and customers belonging to the clusters are input data. These data also include service times for the clusters as a function of the number of deliverymen that can be assigned to a vehicle.

Consider a set of identical vehicles initially located at a central depot. Each can only perform a single route. A solution to the problem consists of a set of routes that start and end at the depot such that the demand of each cluster is served by exactly one vehicle (no splitting allowed) and within the cluster time window. Each route must have a defined number of deliverymen that is maintained throughout the route. Therefore, given a solution \( S \), its cost is defined by Equation (1).

\[
c(S) = w_1v + w_2d + w_3t
\]

where \( v \) is the number of vehicles used, \( d \) is the total number of deliverymen assigned and \( t \) is the total traveled distance in \( S \). Moreover, \( w_1, w_2 \) and \( w_3 \) are the weights of each objective, which can be used to define the priorities of these objectives.

2.1 A set partitioning formulation for the VRPTWMD

The VRPTWMD can be formulated as a set partitioning (SP) model (Munari and Morabito, 2016). Let \( L \) be the maximum number of deliverymen that can be assigned to a single route, \( D \) be the total number of available deliverymen at the depot and \( n \) be the number of customer clusters. Also, let \( P^l \) be the set of all feasible routes using \( l \) deliverymen, \( l = 1, 2, \ldots, L \). We say that a vehicle travels in mode \( l \) if its crew size is equal to \( l \). We associate the following parameters with each route \( p \in P^l \): \( c_p^l \) represents its cost and \( a_{pi}^l \), \( i = 1, \ldots, n \), a binary coefficient taking the value 1 if and only if route \( p \) serves cluster \( i \) in mode \( l \). Finally, let \( \lambda_p^l \) be a binary variable that is equal to 1 if and only if route \( p \) is selected. Using this notation, the VRPTWMD can be formulated as follows:

\[
\min \sum_{l=1}^{L} \sum_{p \in P^l} c_p^l \lambda_p^l
\]
The objective function (2a) minimizes the total cost of the selected routes. Following the definition stated in (1), the cost of each single route \( p \) that sequentially visits the clusters \( i_0, i_1, \ldots, i_r, r > 0 \), is given by Equation (3).

\[
c_l^p = w_1 + w_2 l + w_3 \sum_{j=0}^{r-1} d_{ij} i_{j+1}
\]

where \( d_{ij} \) is the Euclidean distance between clusters \( i \) and \( j \). Constraints (2b) specify that each cluster \( i \) must be visited by exactly one vehicle in one single mode \( l \). Constraints (2c) impose the total number of available deliverymen. Finally, the binary requirements on the variables are imposed by constraints (2d).

A lower bound on the optimal value of (2a)-(2d) can be obtained by solving its linear programming (LP) relaxation. Indeed, the main advantage of this formulation is that it has a stronger LP relaxation than the arc flow formulation proposed by Pureza et al. (2012). Nevertheless, in practice, model (2a)-(2d) may contain a huge number of variables because, in general, the number of routes in sets \( P_l \) is exponential in terms of \( n \). Therefore, the LP relaxation is solved using column generation. Still, because the variables \( \lambda^l_p \) are integers, column generation is embedded in a branch-and-bound algorithm to effectively solve the model (2a)-(2d), leading to a branch-and-price algorithm. Moreover, the LP bounds can be further strengthened by adding valid inequalities to the formulation, yielding a branch-price-and-cut method (Desrosiers and Lübbecke, 2010).

3 Hybrid method

As mentioned previously, interest in hybrid methods combining exact and metaheuristics approaches has grown in recent years, especially for solving routing problems. Taking advantage of their individual components, hybrid methods are usually able to obtain better solutions than those obtained by their components individually as standalone approaches (Jourdan et al., 2009).

A common practice uses heuristic algorithms to populate a column pool with the routes of the local optimal solutions visited during the search phase. Next, a set partitioning or set covering (SC) formulation is solved over the pool to generate the best possible solution through the combination of the routes, typically using a general-purpose integer programming solver. Applications of this type of method can be found in (Kramer et al., 2015; Cacchiani et al., 2014; Yildirim and Çatay, 2014; Villegas et al., 2013; Subramanian et al., 2013; Alvarenga et al., 2007; Russell and Chiang, 2006). These methods can be further extended by incorporating principles of the column generation method (i.e., using dual variables coming from the LP relaxation of the SP/SC formulation). The hybrid method tries to guide the search to enrich the pool, attempting to approach better or optimal solutions (Hauge et al., 2014; Nishi and Izuno, 2014; Parragh and Schmid, 2013; Chen and Xu, 2006). Other hybrid methods include, among others, the use of an integer program to explore promising areas of the search space, which are identified in advance by a heuristic component (Salari et al., 2010; Schmid et al., 2010; Archetti et al., 2008; De Franceschi et al., 2006).
In the proposed hybrid method, we explore the use of two metaheuristic approaches based on ILS and LNS and a BPC algorithm in a cooperative solution approach. The metaheuristics and the BPC interact to improve the ability of the BPC to generate good integer solutions at an earlier stage and therefore to enhance its performance and suitability in practice. The proposed hybrid method is similar to the method used in (Danna and Le Pape, 2005) to solve the VRPTW. The authors develop a hybrid method, combining a branch-and-price algorithm with constructive and local search heuristics. These heuristics aim at helping the branch-and-price algorithm find feasible solutions faster. Nevertheless, one of the main differences between our method and the one proposed by Danna and Le Pape (2005) is that, in the latter, the constructive heuristic is used to build an initial solution and local search algorithms are called at regular time intervals to improve the current incumbent of the method. In contrast, in our method, ILS and LNS are used inside the hybrid method for the following purposes: (i) to generate the columns to initialize the root node of the tree; (ii) to provide the initial incumbent solution of the hybrid method; and (iii) to try to improve each integer solution found by the BPC.

Figure 1 illustrates the proposed hybrid method. The left-hand square represents the BPC algorithm and the right-hand square shows the metaheuristic approaches. The upper and bottom circles represent the restricted master problem and the primal heuristic of the BPC, respectively. The hybrid method functions as follows. First, the ILS and LNS metaheuristics are sequentially executed. The routes of each feasible solution are stored in a column pool, discarding repeated columns. During this phase, the best solution found by the metaheuristics is defined as the initial incumbent of the method. When the column pool reaches a predefined size, the primal heuristic of the BPC is executed over the pool, in an attempt to find a better solution combining these columns. When a new best solution is found, the metaheuristics are called to improve it and the resulting solution is defined as the incumbent solution.

After the first phase, the execution of the BPC is started and when an integer solution is found during its execution, either because the master problem of a node resulted in an integer solution or because the primal heuristic found an integer one, the metaheuristics are sequentially applied to improve it and update the incumbent solution of the hybrid method, if necessary. One single run of $t_{MH}$ seconds of each metaheuristic approach is applied; if one of them improves the solution during a run, they are applied again. This scheme allows us to apply the metaheuristics more often when they succeed in improving the solutions. The hybrid method stops when it reaches a specified maximum running time or when it completes the optimality proof of the incumbent solution within the time limit.

One of the fundamental features underlying this method is the cooperative scheme between the BPC and the metaheuristics. Given that metaheuristics are designed especially to quickly obtain good solutions to the problem, they help the BPC find better integer solutions at an earlier stage. In return, the BPC algorithm helps the metaheuristics by providing different initial solutions for them.

In the remainder of this section we describe the two metaheuristic approaches used in the hybrid method. The first one is based on ILS and is described in Section 3.1. The second one is based on LNS and is described in Section 3.2. The BPC component of the hybrid method is described in Section 3.3. These contents are a review of concepts already described in Álvarez and Munari (2016) and Munari and Morabito (2016), respectively, but are recalled here to make this paper self-contained.

It is worth pointing out that ILS and LNS have been successfully applied to solve many routing
problems (Akpinar, 2016; Silva et al., 2015; Azi et al., 2014; Penna et al., 2013; Hong, 2012; Ropke and Pisinger, 2006). Therefore they are chosen to be incorporated in the developed hybrid method because they show a great potential to successfully cooperate with the BPC algorithm. Moreover, the ILS and LNS of Álvarez and Munari (2016) are the current state-of-the-art metaheuristic methods for the VRPTWMD.

3.1 ILS-based metaheuristic approach

An Iterated Local Search metaheuristic is an algorithm that applies a local search repeatedly over solutions resulting from the perturbation of the visited local optimal, which leads to a randomized walk in the space of local optimal solutions (Lourenço et al., 2003). To devise an ILS algorithm four main parts are needed: an initial solution, a local search procedure for improvement, a perturbation mechanism used to escape from local optimal solutions and an acceptance criterion that determines the solution from which the search should continue.

Our ILS approach combines the local search procedure with a set of additional specific heuristics to iteratively improve the local optimal solution. Due to the use of diverse heuristic components, our approach can tackle different parts of the objective function with the goal of refining the obtained solutions. To this end, the approach comprises two phases. The first phase focuses on reducing the number of vehicles of the solution, perturbing the solutions with a route reduction heuristic. The second phase focuses on reducing the traveled distance, perturbing the solutions with a randomized removal and insertion heuristic. A high-level structure of the approach is given in Algorithm 1. The components of the approach are briefly described below.

For the local search procedure, we use a variable neighborhood descent heuristic (Mladenovic and Hansen, 1997) with random neighborhood ordering (RVND). This heuristic explores increasingly different neighborhoods of the current incumbent solution, repeatedly applying a set of local search operators to reach the local optimal solution. The used RVND relies on six inter-route operators, namely, $\text{Shift}(k,0)$ with $k = 1, 2, 3$; $\text{Swap}(1,1)$; $\text{Swap}(2,1)$; and $\text{Swap}(2,2)$. The $\text{Shift}(k,0)$ operators move $k$ adjacent clusters from one route to another one, while the $\text{Swap}(a,b)$ operators
Algorithm 1: High-level structure of the ILS-based metaheuristic approach.

\[
\text{output: Best solution } S^*; \\
1 \text{ begin } \\
2 \quad S_0 \leftarrow \text{Generate initial solution;} \\
3 \quad S^* \leftarrow \text{Local search}(S_0); \\
4 \quad \text{repeat} \\
5 \quad \quad S' \leftarrow \text{Perturb}(S^*); \\
6 \quad \quad S' \leftarrow \text{Local search}(S'); \\
7 \quad \quad S' \leftarrow \text{Deliverymen reduction}(S'); \\
8 \quad \quad \text{Acceptance criterion }(S', S^*); \\
9 \quad \text{until reaching the stopping criterion;} \\
10 \text{ end }
\]

Next, perturbation operations must be applied to allow the approach to escape from local optimal solutions and explore new regions of the search space. In our approach, one of the perturbation mechanisms consists of one operation of removal and reallocation of up to \( np \) clusters of the current solution. A route reduction heuristic is also applied as perturbation mechanism. This heuristic tries to reallocate all clusters of a given route of the solution into the other routes. Note that this heuristic can both improve the overall quality and completely change the structure of the solution, therefore it was used as improvement heuristic on the approach after each iteration of the RVND heuristic.

The delivery reduction heuristic another improvement heuristic used on the approach. This heuristic is specific for the variable crew size feature of the problem. It tries to reduce the number of deliverymen one route at a time, even if it is necessary to remove clusters of the route. Finally, our acceptance criterion only accepts solutions that improve the local minimum encountered so far.

3.2 LNS-based metaheuristic approach

The other metaheuristic approach in the hybrid method is based on Large Neighborhood Search (Shaw, 1998), a metaheuristic that gradually improves the solutions by successively applying destroy and repair operators. In LNS, a destroy operator removes some components of the current solution using a defined criterion, while the repair operator rebuilds the destroyed solution by reinserting the removed parts using a specified rule. Thereafter an acceptance criterion decides to accept or reject the resulting solution as the new current solution, based on the improvement of the current local minimum. To devise an enhanced solution method, the basic destroy and repair operators are combined with additional specific heuristics for the VRPTWMD, as in the ILS-based approach. A high-level description of the approach is given in Algorithm 2.

In total, the LNS-based approach uses four destroy operators and two repair operators. Each destroy operator removes up to \( q \) clusters of the input solution, returning a partially destroyed solution. The operators used are: random removal, worst removal, related removal and time-
Algorithm 2: High-level structure of the LNS-based metaheuristic approach.

\begin{algorithm}
\textbf{output}: Best solution $S^*$;
\begin{algorithmic}[1]
\State $S_0 \gets$ Generate initial solution;
\State $S^* \gets S_0$;
\Repeat
\Indpunct\State $S' \gets$ Apply destroy and repair operators to $S^+$;
\Indpunct\State $S' \gets$ Route reduction($S'$);
\Indpunct\State $S' \gets$ Deliverymen reduction($S'$);
\Indpunct\State $S' \gets$ RVND($S'$);
\Indpunct\State Acceptance criterion ($S'$, $S^*$);
\Until reaching the stopping criterion;
\State end
\end{algorithmic}
\end{algorithm}

oriented removal. These operators were used by Pisinger and Ropke (2007) for a pickup and delivery problem, and its adaptation to the VRPTWMD is straightforward. The random removal operator selects the clusters to be removed in a random manner. The worst removal operator removes clusters that are expensive (in terms of distance) on their current routes. In the related removal operator, the relatedness of two clusters $i$ and $j$ is measured as the distance between them ($d_{ij}$). In the time-oriented removal operator, relatedness $\Delta_{ij}$ between clusters $i$ and $j$ is measured by $\Delta_{ij} = |w_i - w_j|$, where $w_i$ denotes the time instant in which service starts at cluster $i$. In the latter two operators, closely related clusters are removed because they are expected to be easily exchangeable.

Regarding the repair operators, which are used to restore the feasibility of the destroyed solutions, we use two operators, a greedy insertion heuristic and a regret insertion heuristic, which were also employed by Pisinger and Ropke (2007). The greedy heuristic tries to insert clusters in the cheapest (in terms of additional distance) insertion position, whereas the regret heuristic tries to prioritize the insertion of tightly constrained clusters to anticipate the insertion of customers that have only a few possible insertion positions. Routes reductions and deliverymen reductions are conducted in the same manner as in the ILS-based approach.

3.3 Branch-price-and-cut method

We now briefly describe the branch-price-and-cut (BPC) algorithm that we used to solve the SP model of the VRPTWMD. It was proposed by Munari and Morabito (2016) and uses the interior point BPC framework of Munari and Gondzio (2013, 2015). In a BPC algorithm, lower bounds are computed by solving the LP relaxation of the SP model using column generation. Also, valid inequalities are dynamically added to tighten these bounds.

In the column generation method, we solve the original linear program, called the Master Problem (MP), by employing a reduced version of the problem containing only a subset of variables. This reduced problem is called the Restricted Master Problem (RMP), which, in our application, consists of the LP relaxation of the SP model (2a)-(2d), considering the subsets $\bar{P}_l \subseteq P_l, l =$
1, \ldots, L, as follows:

$$
\begin{align*}
\text{min} & \quad \sum_{l=1}^{L} \sum_{p \in P_l} c^l_p \lambda^l_p \quad (4a) \\
\text{s.a} & \quad \sum_{l=1}^{L} \sum_{p \in P_l} a^l_{pi} \lambda^l_p = 1, \quad i = 1, \ldots, n, \quad (4b) \\
& \quad \sum_{l=1}^{L} \sum_{p \in P_l} \lambda^l_p \leq E, \quad (4c) \\
& \quad \lambda^l_p \geq 0, \quad l = 1, \ldots, L, \quad p \in \bar{P}_l. \quad (4d)
\end{align*}
$$

Let $\lambda^*$ and $(\bar{u}, \bar{v})$ be the optimal primal and dual solutions, respectively, where $\bar{u} \in \mathbb{R}^n$ and $\bar{v} \in \mathbb{R}_-$ are associated to constraints (4b) and (4c), respectively. Using this dual solution, the reduced cost of the $p$th column is given by

$$
\left( c^l_p - \sum_{i=1}^{n} a^l_{pi} \bar{u}_i - l \bar{v} \right). \quad (4c)
$$

At each iteration of the method, we solve the RMP and, using its dual solution, look for a variable/column with negative reduced cost. However, because it is not possible to enumerate all the columns of the sets $P_l, l = 1, \ldots, L$, we must solve an optimization problem (pricing problem), given by (5).

$$
\begin{align*}
\min_{p \in P_l} \left\{ c^l_p - \sum_{i=1}^{n} a^l_{pi} \bar{u}_i - l \bar{v} \right\}, \quad l = 1, \ldots, L. \quad (5)
\end{align*}
$$

In the case of the VRPTWMD, the pricing problem solver is called up to $L$ times at each iteration, once for each mode $l$. These pricing problems are resource constrained shortest path problems and differ from each other in the service times of the clusters associated with the corresponding mode. Each pricing problem is solved through a multi-phase dynamic programming approach as proposed by Desaulniers et al. (2008), which uses a label-setting dynamic programming algorithm further aided by heuristics algorithms. A phase is defined by an ordered sequence of algorithms and parameters associated with them. The algorithms used are: local search operators, heuristic dynamic programming and exact dynamic programming. In a sequence, the algorithms are ordered from the fastest to the slowest, and the last-phase algorithm sequence contains the exact dynamic programming algorithm to guarantee that the LP relaxation of the SP model is solved to its optimality (Munari and Gondzio, 2013).

Subset-row (SR) inequalities (Jepsen et al., 2008) are generally used in BPC algorithms for routing problems, especially SR inequalities based on triplets of nodes as follows. Let $S = \{i_1, i_2, i_3\} \subset \{1, 2, \ldots, n\}$ be a triplet of nodes (clusters of customers). Then the corresponding SR inequality must enforce that at most one route/column visits at least two nodes in $S$. For the VRPTWMD, Munari and Morabito (2016) incorporated the number of deliverymen as follows. Let $I_S \in \bar{P}$ be the subset of routes serving at least two nodes in $S$. The SR inequality is given by Equation (6).

$$
\sum_{l=1}^{L} \sum_{p \in I_S} \lambda^l_p \leq 1 \quad (6)
$$

In the BPC algorithm, the initial set of columns of the restricted master problem in the root node contains all single cluster routes $(0, i, 0), i = 1, \ldots, n$ for each mode $l$, as well as columns generated by a savings algorithm (Clarke and Wright, 1964) and modified through some simple
local search operators. Upper bounds are obtained by a restricted primal heuristic. The heuristic
exploits the fact that any column in a given restricted master problem corresponds to a feasible
route and, therefore, one can try to obtain a feasible solution to the problem from the combination
of a subset of these routes. Then, after completing the column generation process, the heuristic
imposes integrality on the master variables and solves the integer programming problem using a
general-purpose MIP solver. Each single call to the primal heuristic runs for up to $t_{SP}$ seconds,
because solving the resulting MIP to optimality can be very time consuming.

The search tree is explored using a best-first strategy, that is, the node with the lowest lower
bound is the node to be processed next. Branching is performed in the tree using three rules.
They are presented in order of priority and, independently of the rule applied, two new child nodes
are created. The first branching rule branches on the value of the number of vehicles used in the
fractional solution. When this value is an integer, the second rule calculates the total number of
deliverymen used in the master problem and branches on that. Finally, the third branching rule
is on the flow of an arc if both the number of vehicles and deliverymen of the fractional solution
are integer. To improve the branching strategy, the BPC algorithm uses the strong branching
technique (Achterberg et al., 2005; Santos et al., 2015). Furthermore, the branching process is
performed using the early branching approach described by Munari and Gondzio (2013).

The computational implementation uses the stabilization technique known as the primal-dual
column generation method (Gondzio et al., 2013; Munari et al., 2011), which employs an interior
point algorithm to solve the restricted master problems in the column generation process. This
algorithm can provide well-centered suboptimal solutions without requiring any artificial resource.
Due to this centrality, the oscillation of the dual solutions is relatively small from one iteration to
another, which contributes to improve the performance of the column generation method regarding
the number of iterations and the running time. Moreover, the central primal solutions provided by
the interior point algorithm can also be used to reduce the number of generated valid inequalities
because deeper cuts can be separated. These cuts improve the lower bound provided by the master
problem more quickly than the cuts generated from optimal solutions (Munari and Gondzio, 2013).
In the BPC algorithm, the valid inequalities are separated within the column generation process.
For a detailed description and extensive computational experiments with the BPC algorithm see
Munari and Morabito (2016).

4 Computational experiments

In this section, we present the results of our computational experiments. First, we describe the
instances used in our tests. Next, we report the results obtained with the hybrid method and
compare its performance against standalone versions of the BPC and the metaheuristics. Finally,
we perform some additional analyses regarding the characteristics of the problem.

In tables presented hereafter, Inst denotes the name of the instance, Cost represents the cost
of the solution found, Veh is the number of vehicles used in the solution, Dist indicates the total
distance covered by the solution, Del denotes the total number of deliverymen allocated to the
solution, Nodes corresponds to the number of nodes processed in the search tree, Total time shows
the time used to solve the instance (in seconds) and Time best indicates the time required to
find the best incumbent of the method (in seconds). Optimal solutions are marked with a '*' to
the right of their cost. For each instance/class, bold text shows the best result (in terms of the
cost of the solution). As the cost is presented using only two decimal places, ties are broken by
the following rules, presented in descending order of priority: proven optimality; shorter traveled
distance; shorter total time; and shorter time to best solution.

All tests with the hybrid method were performed on an Intel(R) Core(TM) i7-2600 3.4 GHz processor and 16 GB RAM. The algorithms were implemented in C++ and the primal heuristic of the BPC uses the IBM ILOG Cplex Optimizer version 12.4 with its default parameter settings.

The parameters of the problem were defined with the same values used in (Pureza et al., 2012; Senarcens and Reimann, 2014; Munari and Morabito, 2016; Álvarez and Munari, 2016) to maintain the same structure used in those papers. Thus, we assume that the maximum number of deliverymen allowed in a single vehicle is $L = 3$ (limited by space restrictions in the cab of an average truck). We also assume that the total number of deliverymen is sufficiently large for any instance ($D = 50$), that travel costs are equal to the distances and the weights of the objective function are $w_1 = 1$, $w_2 = 0.1$ and $w_3 = 0.0001$. Note that the chosen weights prioritize solutions using fewer vehicles, followed by those using fewer deliverymen and, finally, solutions with shorter traveled distances.

The values adopted for the parameters of the metaheuristic approaches are the same used by Álvarez and Munari (2016), which were defined through a design of experiments (Montgomery, 2008). The time limit for the primal heuristic of the BPC algorithm and the metaheuristics in the hybrid method were defined through preliminary experiments as $t_{SP} = 30$ and $t_{MH} = 5$.

4.1 Test instances

To evaluate our hybrid method, we used the well-known benchmark instances proposed by Solomon (1987) modified for the VRPTWMD. The set of 56 instances, each one with 100 nodes, is divided into three classes according to the geographical distribution of customers: randomly distributed (R), clustered (C) and a mix of randomly distributed and clustered (RC). Each class is further divided into two classes. Classes R1, C1 and RC1 are characterized by vehicles with smaller capacity and customers with tighter time windows than those in classes R2, C2 and RC2. Note that because of their characteristics, instances of the classes 2 are more challenging for column generation-based methods (Desaulniers et al., 2014).

Following Pureza et al. (2012), we assume that the coordinates of the customers in the Solomon’s instances are the coordinates for the parking location of the cluster in the VRPTWMD. Also, all the original data of the instances is kept, except for the service time at each node, which is replaced by the service time at the node according to a mode $l$, computed by Equation (7).

$$s_{il} = \min\{rs \times d_i, w_0^b - \max\{w_i^a, d_0\} - d_{i0}\}/l, \quad i = 1, \ldots, n, \quad l = 1, \ldots, L$$  \hspace{1cm} (7)

where $rs$ is the service rate (in the experiments we used $rs = 2$) and $d_i$ is the demand of the node $i$. The second term in the equation guarantees the feasibility of the instance.

4.2 Comparison of the hybrid method against the standalone BPC method

In this section we compare the performance of the hybrid method with the BPC algorithm used as a standalone method. Tables 1-6 show the results obtained by these methods within a one-hour time limit. In these tables, columns 2-7 and 8-13 present the best solution (and its attributes) found by the BPC algorithm and the hybrid method, respectively. The last row of the tables (Avg) shows the arithmetic mean value of each column.

The analysis of these results reveals that for instances in the classes 1 (Tables 1-3) there are relatively small cost differences between the best solutions found because the structure of these
<table>
<thead>
<tr>
<th>Inst</th>
<th>Cost</th>
<th>Veh</th>
<th>Dist</th>
<th>Del</th>
<th>Nodes</th>
<th>Time</th>
<th>Total Cost</th>
<th>Cost</th>
<th>Veh</th>
<th>Dist</th>
<th>Del</th>
<th>Nodes</th>
<th>Time</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>R101</td>
<td>23.67</td>
<td>19</td>
<td>1720.11</td>
<td>45</td>
<td>3</td>
<td>26</td>
<td>29</td>
<td>23.67</td>
<td>19</td>
<td>1720.11</td>
<td>45</td>
<td>3</td>
<td>26</td>
<td>29</td>
</tr>
<tr>
<td>R102</td>
<td>20.95</td>
<td>17</td>
<td>1530.92</td>
<td>38</td>
<td>1</td>
<td>14</td>
<td>15</td>
<td>20.95</td>
<td>17</td>
<td>1530.92</td>
<td>38</td>
<td>1</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>R103</td>
<td>15.84</td>
<td>13</td>
<td>1369.22</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R104</td>
<td>13.62</td>
<td>11</td>
<td>1164.77</td>
<td>25</td>
<td>17</td>
<td>2127</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R105</td>
<td>17.66</td>
<td>14</td>
<td>1594.71</td>
<td>35</td>
<td>81</td>
<td>2677</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R106</td>
<td>14.34</td>
<td>11</td>
<td>1382.93</td>
<td>32</td>
<td>50</td>
<td>2498</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R107</td>
<td>12.82</td>
<td>10</td>
<td>1182.93</td>
<td>32</td>
<td>50</td>
<td>2498</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R108</td>
<td>11.60</td>
<td>9</td>
<td>1000.74</td>
<td>25</td>
<td>17</td>
<td>2127</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R109</td>
<td>13.62</td>
<td>11</td>
<td>1164.77</td>
<td>25</td>
<td>17</td>
<td>2127</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R110</td>
<td>17.66</td>
<td>14</td>
<td>1594.71</td>
<td>35</td>
<td>81</td>
<td>2677</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R111</td>
<td>14.34</td>
<td>11</td>
<td>1382.93</td>
<td>32</td>
<td>50</td>
<td>2498</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R112</td>
<td>12.82</td>
<td>10</td>
<td>1182.93</td>
<td>32</td>
<td>50</td>
<td>2498</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>R113</td>
<td>11.60</td>
<td>9</td>
<td>1000.74</td>
<td>25</td>
<td>17</td>
<td>2127</td>
<td>3600</td>
<td>15.84</td>
<td>13</td>
<td>1368.73</td>
<td>27</td>
<td>32</td>
<td>2130</td>
<td>3600</td>
</tr>
<tr>
<td>Avg.</td>
<td>15.28</td>
<td>12.08</td>
<td>1303.75</td>
<td>30.67</td>
<td>29.00</td>
<td>1872.83</td>
<td>3003.66</td>
<td>15.17</td>
<td>12.00</td>
<td>1272.09</td>
<td>30.42</td>
<td>17.25</td>
<td>1426.17</td>
<td>3008.40</td>
</tr>
</tbody>
</table>

Tab. 1: BPC algorithm vs hybrid method results for the instances in class R1.

instances allows the BPC algorithm to obtain good feasible solutions. In terms of running times, note that for classes RC1 and C1 the hybrid method was faster when both algorithms found the optimal solution within the time limit. It is also worth pointing out that in class C1 the hybrid method was almost 3 times faster than the BPC when both methods found the optimal solution, leading to a significant performance improvement compared to the BPC algorithm. Moreover, for the instance C104, the hybrid method found a better solution (due to smaller total distance). Nevertheless, for instances R101 and R102, which are considered to be easy, the BPC algorithm was faster than the hybrid method because of the time the hybrid method spent in the initial phase (metaheuristics and primal heuristic call).

For instances not solved to optimality in classes R1 and RC1, the average number of nodes

<table>
<thead>
<tr>
<th>Inst</th>
<th>Cost</th>
<th>Veh</th>
<th>Dist</th>
<th>Del</th>
<th>Nodes</th>
<th>Time</th>
<th>Total Cost</th>
<th>Cost</th>
<th>Veh</th>
<th>Dist</th>
<th>Del</th>
<th>Nodes</th>
<th>Time</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC101</td>
<td>20.19</td>
<td>16</td>
<td>1862.21</td>
<td>40</td>
<td>97</td>
<td>1453</td>
<td>3600</td>
<td>20.19</td>
<td>16</td>
<td>1862.21</td>
<td>40</td>
<td>97</td>
<td>1453</td>
<td>3600</td>
</tr>
<tr>
<td>RC102</td>
<td>17.08</td>
<td>13</td>
<td>1777.29</td>
<td>39</td>
<td>21</td>
<td>613</td>
<td>848</td>
<td>17.08</td>
<td>13</td>
<td>1777.29</td>
<td>39</td>
<td>21</td>
<td>613</td>
<td>848</td>
</tr>
<tr>
<td>RC103</td>
<td>14.44</td>
<td>11</td>
<td>1393.87</td>
<td>33</td>
<td>33</td>
<td>1832</td>
<td>3600</td>
<td>14.44</td>
<td>11</td>
<td>1393.87</td>
<td>33</td>
<td>33</td>
<td>1832</td>
<td>3600</td>
</tr>
<tr>
<td>RC104</td>
<td>14.03</td>
<td>11</td>
<td>1299.06</td>
<td>29</td>
<td>8</td>
<td>846</td>
<td>3600</td>
<td>14.02</td>
<td>11</td>
<td>1299.06</td>
<td>29</td>
<td>8</td>
<td>846</td>
<td>3600</td>
</tr>
<tr>
<td>RC105</td>
<td>17.87</td>
<td>14</td>
<td>1724.32</td>
<td>37</td>
<td>65</td>
<td>3120</td>
<td>3600</td>
<td>17.87</td>
<td>14</td>
<td>1724.32</td>
<td>37</td>
<td>65</td>
<td>3120</td>
<td>3600</td>
</tr>
<tr>
<td>RC106</td>
<td>16.66</td>
<td>13</td>
<td>1641.94</td>
<td>35</td>
<td>35</td>
<td>3757</td>
<td>3600</td>
<td>16.65</td>
<td>13</td>
<td>1641.94</td>
<td>35</td>
<td>35</td>
<td>3757</td>
<td>3600</td>
</tr>
<tr>
<td>RC107</td>
<td>14.33</td>
<td>11</td>
<td>1383.18</td>
<td>32</td>
<td>8</td>
<td>3319</td>
<td>3600</td>
<td>14.33</td>
<td>11</td>
<td>1383.18</td>
<td>32</td>
<td>8</td>
<td>3319</td>
<td>3600</td>
</tr>
<tr>
<td>RC108</td>
<td>14.03</td>
<td>11</td>
<td>1288.03</td>
<td>29</td>
<td>5</td>
<td>908</td>
<td>3600</td>
<td>14.03</td>
<td>11</td>
<td>1288.03</td>
<td>29</td>
<td>5</td>
<td>908</td>
<td>3600</td>
</tr>
<tr>
<td>Avg.</td>
<td>16.08</td>
<td>12.50</td>
<td>1540.61</td>
<td>34.25</td>
<td>34.00</td>
<td>1958.25</td>
<td>3185.46</td>
<td>16.08</td>
<td>12.50</td>
<td>1518.48</td>
<td>34.25</td>
<td>23.88</td>
<td>1289.50</td>
<td>3062.32</td>
</tr>
</tbody>
</table>

Tab. 2: BPC algorithm vs hybrid method results for the instances in class RC1.

Tab. 3: BPC algorithm vs hybrid method results for the instances in class C1.
### Computational experiments

<table>
<thead>
<tr>
<th>Inst</th>
<th>BPC algorithm</th>
<th>Hybrid method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>Veh</td>
<td>Dist</td>
</tr>
<tr>
<td>best</td>
<td>time</td>
<td>best</td>
</tr>
<tr>
<td>R201</td>
<td>5.36</td>
<td>4</td>
</tr>
<tr>
<td>R202</td>
<td>4.95</td>
<td>4</td>
</tr>
<tr>
<td>R203</td>
<td>4.04</td>
<td>3</td>
</tr>
<tr>
<td>R204</td>
<td>4.01</td>
<td>3</td>
</tr>
<tr>
<td>R205</td>
<td>4.94</td>
<td>4</td>
</tr>
<tr>
<td>R206</td>
<td>4.02</td>
<td>3</td>
</tr>
<tr>
<td>R207</td>
<td>3.71</td>
<td>3</td>
</tr>
<tr>
<td>R208</td>
<td>3.70</td>
<td>3</td>
</tr>
<tr>
<td>R209</td>
<td>4.93</td>
<td>4</td>
</tr>
<tr>
<td>R210</td>
<td>4.03</td>
<td>3</td>
</tr>
<tr>
<td>R211</td>
<td>3.70</td>
<td>3</td>
</tr>
</tbody>
</table>

**Avg.** 4.31 3.36 1264.70 8.18 2.18 0.09 3600.00 3.79 3.09 1011.79 6.00 1.82 716.36 3600.00

**Tab. 4:** BPC algorithm vs hybrid method results for the instances in class R2.

<table>
<thead>
<tr>
<th>Inst</th>
<th>BPC algorithm</th>
<th>Hybrid method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>Veh</td>
<td>Dist</td>
</tr>
<tr>
<td>best</td>
<td>time</td>
<td>best</td>
</tr>
<tr>
<td>RC201</td>
<td>6.68</td>
<td>5</td>
</tr>
<tr>
<td>RC202</td>
<td>5.37</td>
<td>4</td>
</tr>
<tr>
<td>RC203</td>
<td>4.96</td>
<td>4</td>
</tr>
<tr>
<td>RC204</td>
<td>4.93</td>
<td>4</td>
</tr>
<tr>
<td>RC205</td>
<td>6.21</td>
<td>5</td>
</tr>
<tr>
<td>RC206</td>
<td>5.35</td>
<td>4</td>
</tr>
<tr>
<td>RC207</td>
<td>5.36</td>
<td>4</td>
</tr>
<tr>
<td>RC208</td>
<td>4.02</td>
<td>3</td>
</tr>
</tbody>
</table>

**Avg.** 5.36 4.13 1600.47 10.75 4.13 0.13 3600.00 4.33 3.50 1198.30 8.13 4.25 1595.75 3600.00

**Tab. 5:** BPC algorithm vs hybrid method results for the instances in class RC2.

<table>
<thead>
<tr>
<th>Inst</th>
<th>BPC algorithm</th>
<th>Hybrid method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>Veh</td>
<td>Dist</td>
</tr>
<tr>
<td>best</td>
<td>time</td>
<td>best</td>
</tr>
<tr>
<td>C201</td>
<td>3.36*</td>
<td>3</td>
</tr>
<tr>
<td>C202</td>
<td>3.36</td>
<td>3</td>
</tr>
<tr>
<td>C203</td>
<td>3.36</td>
<td>3</td>
</tr>
<tr>
<td>C204</td>
<td>3.36</td>
<td>3</td>
</tr>
<tr>
<td>C205</td>
<td>3.36*</td>
<td>3</td>
</tr>
<tr>
<td>C206</td>
<td>3.36*</td>
<td>3</td>
</tr>
<tr>
<td>C207</td>
<td>3.36</td>
<td>3</td>
</tr>
<tr>
<td>C208</td>
<td>3.36*</td>
<td>3</td>
</tr>
</tbody>
</table>

**Avg.** 3.36 3.00 587.51 3.00 1.00 2112.13 2457.05 3.36 3.00 587.51 3.00 1.00 28.50 2005.46

**Tab. 6:** BPC algorithm vs hybrid method results for the instances in class C2.
explored in the hybrid method is reduced compared to the BPC algorithm. This is because the hybrid method spends time on metaheuristics, which reduces the time for running the column generation method and the other components of the BPC. Finally, with respect to the time required to find the final incumbent (Time best), it can be observed that the hybrid method reduced the average time 23% and 34% for classes R1 and RC1, respectively, compared to the BPC algorithm. Note that for all but one instance of class C1, the hybrid method quickly found the optimal solution and spent the rest of the time trying to prove optimality.

On the other hand, for instances in the classes 2 (Tables 4-6) the results show substantial gains in terms of the cost of the best solutions found. The major differences are obtained for classes R2 and RC2, in which the proposed hybrid method reduced the average cost compared to the BPC algorithm by 12% and 17%, respectively. For class C2, both methods found the same solutions for all instances. Nevertheless, for this last class, the hybrid method was, on average, more than two times faster than the BPC algorithm when both methods proved the optimality of the solution found. Additionally, the hybrid method proved the optimality of one solution (C207) that the BPC could not prove. Both methods explore a similar number of nodes in the search tree. It is worth noting that, in these classes, the average number of nodes explored is low with respect to classes 1.

Regarding the time to reach the final incumbent, in classes R2 and RC2 the BPC algorithm found a first solution by calling the primal heuristic on the initial columns, which are generated by some simple heuristics (see Section 3.3), and these solutions remained as the incumbents until the algorithm stopped. The hybrid method used more time to reach the final incumbents, but it found much better solutions than those found by the BPC algorithm. Similar to class C1, in class C2 the hybrid method rapidly found the optimal solution of five instances and spent the rest of the time proving its optimality.

In summary, these results show the advantages of using specific algorithms to obtain good feasible solutions within an exact method. Note that the instances in classes 2 are more challenging that those in classes 1, as only a few of them were solved to optimality within one hour as well as only few nodes were explored within this time limit.

4.3 Comparison of the hybrid method against the standalone metaheuristics

We now compare the results of the hybrid method with respect to standalone versions of the metaheuristic approaches, using relatively short running times. This analysis is made to assess the performance of the hybrid method with respect to specialized methods developed to quickly provide good solutions. We compare the results reported in Álvarez and Munari (2016) to a single execution of the hybrid method using a time limit of 10 minutes. Table 7 shows the average costs for each instance class. It can be observed that the hybrid method can be competitive with respect to the metaheuristics for the classes 1 because the relative difference between the average cost of the solutions is relatively small for class R1, and the hybrid method clearly outperforms the metaheuristics for class RC1. The classes R2 and RC2 are more difficult for the hybrid method; therefore, the metaheuristics performed better on these two classes. Finally, note that in classes C1 and C2 all methods found the same solutions, but the running times of the hybrid method are lower because it can prove the optimality of the solutions and stop early.
4.4 Source of the integer solutions in the methods

In this section, we show the sources of the final integer solutions in the hybrid method and the BPC algorithm. Notice that these methods can have up to four sources of integer solutions, namely, (i) initial call of the metaheuristics; (ii) restricted master problem (RMP); (iii) primal heuristic (SP); and (iv) metaheuristic algorithms. Remember that the last source (metaheuristics algorithms) uses the solutions of the primal heuristic and the integer solutions of the restricted master problem as starting solutions. Also note that the first and last sources are only present in the hybrid method. A distinction between the first and subsequent calls to the metaheuristic algorithms as sources of integer solutions was made to assess the capacity of the metaheuristics to generate good feasible solutions in very short running times (first source). Table 8 reports the number of times that each component found the best final solution under time limits of both 600 and 3,600 seconds.

<table>
<thead>
<tr>
<th>Source</th>
<th>BPC 600 s</th>
<th>BPC 3600 s</th>
<th>Hybrid 600 s</th>
<th>Hybrid 3600 s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial call of the metaheuristics</td>
<td>- -</td>
<td>- -</td>
<td>22</td>
<td>26</td>
</tr>
<tr>
<td>Restricted master problem</td>
<td>11</td>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Primal heuristic</td>
<td>45</td>
<td>43</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Metaheuristics algorithms</td>
<td>- -</td>
<td>- -</td>
<td>32</td>
<td>28</td>
</tr>
</tbody>
</table>

These results reveal that the BPC algorithm strongly depends on the primal heuristic to find integer solutions because the primal heuristic provides more than 75% of them. The remaining solutions of the BPC algorithm are provided by the restricted master problem and correspond to the solutions for the instances in classes C1 and C2 that were solved to optimality within the time limit.

For the hybrid method, most of the solutions are provided by its metaheuristic component, especially when the metaheuristics use solutions coming from the primal heuristic and the restricted master problem of the BPC. Note that only few of the final incumbent solutions come from the sources RMP and SP. This does not mean that these components are not capable of generating feasible solutions, but rather that the metaheuristics can improve them in most cases. These results confirm that the developed cooperative scheme succeeds in enhancing the capacity of the
BPC algorithm to find good feasible solutions.

4.5 Additional analyses

In this section, we first analyze the route structures of the problem in terms of the times that compose them. To this end, we analyzed the solutions of the hybrid method (under a one-hour time limit), to identify the proportions of the total time corresponding to service, travel and waiting. Note that waiting times occur when a vehicle arrives at a cluster before its time window opens. The aggregated results for each instance class are shown in Table 9, which shows that the total times of the delivery routes are composed mainly of service times (46% on average) followed by travel times (32% on average). These results highlight the relevance of considering the service time as dependent on the number of deliverymen and, in turn, the need to consider the number of deliverymen as a decision variable. This structure of the times in the delivery routes can be observed in real-life applications, especially in the distribution of beverage and dairy products in highly congested urban areas. In those situations, service times can account for up to 50% of the total route times. This fact is one of the main motivations for the study of the VRPTWMD (Pureza et al., 2012).

Also note that the times for the solutions of the classes C1 and C2 are dominated by waiting times, because in these instances the clusters are grouped in geographical space and, hence, vehicles spend more time waiting at the clusters than they do traveling between the parking locations for the clusters.

<table>
<thead>
<tr>
<th>Class</th>
<th>Time</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Travel</td>
<td>Service</td>
<td>Waiting</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>49%</td>
<td>46%</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>RC1</td>
<td>53%</td>
<td>44%</td>
<td>3%</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>10%</td>
<td>45%</td>
<td>45%</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>36%</td>
<td>52%</td>
<td>12%</td>
<td></td>
</tr>
<tr>
<td>RC2</td>
<td>40%</td>
<td>48%</td>
<td>12%</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>7%</td>
<td>41%</td>
<td>52%</td>
<td></td>
</tr>
<tr>
<td>Avg</td>
<td>32%</td>
<td>46%</td>
<td>22%</td>
<td></td>
</tr>
</tbody>
</table>

Tab. 9: Composition of times in the routes of the different instance classes.

Next, we analyzed scenarios that impose different limits on the numbers of deliverymen on board, as in practice it may be valuable to present such solution alternatives. To perform these tests, we used the hybrid method, varying the maximum number of deliverymen allowed on board between one and four. Remember that Pureza et al. (2012) established a limit of three deliverymen on board due to the cab capacity restriction of an average truck. To evaluate these scenarios, we used the instances in class R1, enforcing a time limit of one hour. The results are presented in Table 10.

It can be observed that increasing the limit of the crew size on board also increases the total number of deliverymen used, on average. However, this increase allows the solution to use a smaller number of vehicles and, therefore, reduces the average cost of the solutions. This result highlights the compromise between the costs involved, because it is beneficial to invest in additional deliverymen to reduce the service times, enabling the use of fewer vehicles (whose cost is higher) and hence to reduce the total cost of the system. It is worth mentioning that for most instances of class R1 the hybrid method stopped without proving optimality due to the imposed time limit; therefore, the obtained solutions can be non-optimal and, consequently, the results of these analyses
5 Conclusions

This paper presented a hybrid method for the VRPTWMD that combines a branch-price-and-cut algorithm with two metaheuristic approaches. The proposed cooperative scheme aims at improving the ability of the BPC algorithm to generate good integer solutions at an earlier stage. The reported computational results show that the developed method outperforms the BPC algorithm used as a standalone method; in particular, it is faster in most instances that are solved to optimality by both methods. Moreover, for instances that are not solved to optimality, the hybrid method provides better feasible solutions. In summary, the results of this paper reveal that using fast heuristic algorithms within an exact method is a promising line of research that is extensible to other vehicle routing problems. Additional analyses were performed to show the importance of considering the number of deliverymen assigned to delivery routes as a decision variable together with routing and scheduling decisions.

In future research we suggest adapting the metaheuristic algorithms to solve the pricing problem and incorporate them into the column generation method trying to speed-up the BPC algorithm and, therefore, to speed-up the hybrid method as well. In addition, the hybrid method can be adapted to tackle several VRPTWMD variants that consider other practical characteristics such as multiple depots, simultaneous pickup and delivery and a heterogeneous fleet. Finally, it would be interesting to take uncertainty into account in the context of multiple deliverymen.

Acknowledgments

This research was partially supported by CAPES-DS (Coordination for the Improvement of Higher Education Personnel, Brazil), FAPESP (Sao Paulo Research Foundation, Brazil, project 2014/00939-8) and CNPq (National Counsel of Technological and Scientific Development, Brazil, project 482664/2013-4).
References


