European Journal of Operational Research 214 (2011) 536–545

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model

Rita Macedo [a], Cláudio Alves [a,*], J.M. Valério de Carvalho [a], François Clautiaux [b], Saïd Hanafi [c]

[a] Centro de Investigação Algoritmi da Universidade do Minho, Escola de Engenharia, Universidade do Minho, 4710-057 Braga, Portugal
[b] Université des Sciences et Technologies de Lille, LIFL UMR CNRS 8022, INRIA Bâtiment INRIA, Parc de la Haute Borne, 59655 Villeneuve d'Ascq, France
[c] LAMIH-SIADE, UMR 8530, Université de Valenciennes et du Hainaut-Cambrésis, Le Mont Houy, 59313 Valenciennes Cedex 9, France

ARTICLE INFO

Article history:
Received 22 October 2010
Accepted 27 April 2011
Available online 4 May 2011

Keywords:
Integer programming
Combinatorial optimization
Routing
Network flows

ABSTRACT

In this paper, we address a variant of the vehicle routing problem called the vehicle routing problem with time windows and multiple routes. It considers that a given vehicle can be assigned to more than one route per planning period. We propose a new exact algorithm for this problem. Our algorithm is iterative and it relies on a pseudo-polynomial network flow model whose nodes represent time instants, and whose arcs represent feasible vehicle routes. This algorithm was tested on a set of benchmark instances from the literature. The computational results show that our method is able to solve more instances than the only other exact method described so far in the literature, and it clearly outperforms this method in terms of computing time.

1. Introduction

The vehicle routing problem (VRP) is a combinatorial optimization problem that has been widely studied in the literature, ever since it was formulated for the first time in [9], and later in [7]. Generally speaking, it is the problem of scheduling a fleet of vehicles to visit a set of customers, to whom they must deliver or from whom they must collect a demanded quantity of goods. The problem consists of finding the best set of routes, according to a given objective function, such that all operational constraints of the vehicles are respected, and the set of customers is covered. This objective function can be the minimization of all traveling costs, the maximization of the number of served customers, or some combination of these or other factors. It can be seen as a generalization of another well known combinatorial problem, the traveling salesman problem, which can be described as a VRP with one vehicle, no depot, no vehicle capacities and no customer demands. The VRP is well-known to be NP-hard, and so are most of its variants. Its solution methods include several heuristic and metaheuristic approaches, as well as some exact methods, mainly based on branch-and-bound techniques. The classical version of the VRP is commonly called the Capacitated vehicle routing problem, as the vehicles in the fleet have limited capacities. There are several variants of this problem. In [8,21], the authors describe the VRP and some of its main variants, like the Pickup and Delivery VRP, the VRP with Backhauls, the Multi Period VRP or the Split Delivery VRP. Recently, the combination of the VRP with other problems also deserved some attention. For example, some authors combined the problem of selecting routes with the problem of loading the vehicles that perform them [10,12–14].

A very known and studied variant of the classical VRP is the VRP with time windows. For this problem, there is a time period associated to each customer, and also to the depot. All customers must be served within their established period of time, and all routes must be performed within the depot's time window. Along with the distances between all customers and depot, it is also necessary to consider traveling and service times. Surveys on the vehicle routing problem with time windows are presented in [5,6], where the authors describe heuristic and metaheuristic solution methods from the literature. Another interesting variant of the VRP has been approached by some authors. It considers that a vehicle can be assigned to more than one route per planning period and has been denoted as the Multi Trip vehicle routing problem or vehicle routing problem with multiple routes. It was first approached in [11]. Some heuristic solution methods [1,4,15–17,20] are described in the survey provided in [18]. All these main variants can be combined with further versions of the problem. Just to state a few, there can be multiple or single depots, homogeneous or heterogeneous fleets, customers can have stochastic or deterministic demands, the problem can be static or dynamic. In this paper, we address the vehicle routing problem with time windows and multiple routes (MVRPTW). Despite its apparent practical relevance (delivering perishable goods, for example), this variant of the classical VRP has not been the subject of a large number of studies. In what concerns exact methods, to our knowledge there is only one contribu-

* Corresponding author. Tel.: +351 253 604765; fax: +351 253604741.
E-mail addresses: rita@dps.uminho.pt (R. Macedo), claudio@dps.uminho.pt (C. Alves), vc@dps.uminho.pt (J.M. Valério de Carvalho), francois.clautiaux@univ-lille1.fr (F. Clautiaux), said.hanafi@univ-valenciennes.fr (S. Hanafi).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.04.037

tion in the literature [3]. It is a generalization of a previous method [2], which considered the same problem but with a single vehicle available. It is a branch-and-price algorithm, with a master problem that is a set-covering problem with variables representing workdays, i.e., sequences of routes assigned to one vehicle for one planning period. The pricing problem is an elementary shortest path problem with resource constraints, formulated in a graph whose nodes represent vehicle routes. The vehicle routes are generated a priori. This is possible because there is an additional constraint in what concerns their duration, which makes the number of feasible routes decrease drastically. With this exact solution method, the authors were able to solve instances with up to 40 customers.

In this paper, we present a new exact solution approach for the MVRPTW. As in [3], we consider the additional route duration constraint and generate all feasible vehicle routes a priori. We propose a new algorithm that is based on a pseudo-polynomial network flow model, whose nodes represent discrete time instants and whose solution is composed of a set of paths, each representing a workday. An issue of this model is that its size depends on the duration of the workdays. The time instants we consider in the model are integer, and so, when non integer traveling times occur, we use rounding procedures that allow us to obtain a (strong) lower bound. Our model is then embedded in an exact algorithm that iteratively adds new time instants to the network flow model, and reoptimizes it, until the solution found is proved to be feasible. Practically speaking, the number of iterations is generally rather small (one in most of the cases). We tested our algorithm on the benchmark used in [3] to compare our results with theirs. Our method outperforms the column-generation based algorithm of [3] in many cases. In the cases where the two methods find a solution, our method drastically reduces the computational time needed.

The paper is organized as follows. In Section 2, we define our problem, along with its notation, and briefly recall some integer programming models from the literature. We contribute to the resolution of this problem with a new network flow formulation, where variables represent feasible vehicles routes. This model is described in Section 3, as well as some reduction criteria to eliminate some of the vehicle routes, in order to improve its efficiency. The network flow model is embedded in an exact solution algorithm that is thoroughly described in Section 4. This algorithm was tested on a set of benchmark instances from the literature. In Section 5 we present the results of those computational tests. Finally, some conclusions are drawn in Section 6.

2. Integer programming models for the MVRPTW

In this Section, we define the problem we are addressing and briefly describe two different formulations from the literature, both approached in [3]. The first one is a compact formulation for this problem and the second one is a set partitioning formulation, corresponding to the master problem of a column generation framework proposed by the authors.

2.1. Problem definition and notation

In the MVRPTW, there is a single depot, denoted by o, which is the beginning and the end of all the vehicle routes. The fleet of vehicles is homogeneous. All the vehicles have a capacity of Q units. It is assumed that there are K available vehicles in the fleet.

The set of customers is represented by $N = \{1,\dots,n\}$. There is a distance, d_{ij}, and a traveling time, t_{ij}, associated to every pair i, $j \in N \cup \{o\}$. Each customer i has a demand q_i, a revenue g_i, a service time s_i and a time window $[a_i,b_i]$, where a_i is the earliest time and b_i the latest time to start the service at customer i. This means that if a vehicle arrives at customer i earlier than a_i, it must wait. We assume, without loss of generality, that a vehicle starts the service at a customer as soon as possible. The service time for the depot is defined as $s_o = 0$, and all the vehicle routes must respect the depot's time window, $[a_o,b_o]$, which means that no vehicle can leave the depot before a_o, nor access it after b_o. This time window represents the duration W of a workday. We assume that $b_i + s_i + d_{i0} \leqslant b_0$, $\forall i \in N$.

Each vehicle can perform several routes during a workday. It means that it can perform one route, reload at the depot and leave to the following route, until the end of the workday. A route r is defined by a sequence of visits to a subset of customers $N_r \subseteq N$. It is feasible if the sum of the demands of all customers that belong to N_r does not exceed the vehicle capacity and if its sequence of visits is such that it is possible to visit every customer within its time window. We also consider that the service of all customers in the route cannot start later that t_{max} time units after the route begins. We denote by R the set of all feasible routes. For each route, there is also a setup time to consider. Before leaving the depot to perform route r, the vehicle needs $\beta\sum_{i\in N_r} s_i$ time units to load, with $\beta \in \mathbb{R}^+$.

Note that it may not be possible to visit all customers due to the limitation on the number of available vehicles. However, it is always desirable to visit as many customers as possible.

2.2. A compact formulation for the MVRPTW

The problem can be formulated in a complete directed graph $G = (V,A)$, being $V = N \cup \{o\}$ its set of nodes and $A = \{(i,j) : i,j \in V\}$ its set of arcs. This compact formulation, where binary variables assign customers to routes and define consecutive pairs of routes, is proposed in [3]. Its binary variables x_{ij}^r and y_i^r define, respectively, if arc (i,j) and customer i belong to route r, whereas the binary variables z_{rs} define if there is a vehicle that performs route r followed by route s in its workday. Notation $r < s$ means that a same vehicle is assigned to perform route s after having performed route r. Variables t_i^r represent the starting instant of service at customer i, if it is served by route r, and t_o^r and $t_o^{r'}$ represent the starting and ending times of route r, respectively. Let M be a sufficiently large number. This formulation states as follows.

$$\min \sum_{r\in R}\sum_{(i,j)\in A} d_{ij}x_{ij}^r - \alpha\sum_{r\in R}\sum_{i\in N} g_i y_i^r \tag{1}$$

$$\text{s.t.} \sum_{j\in V} x_{ij}^r = y_i^r, \quad \forall i \in N, \quad \forall r \in R, \tag{2}$$

$$\sum_{r\in R} y_i^r \leqslant 1, \quad \forall i \in N, \tag{3}$$

$$\sum_{i\in V} x_{ih}^r - \sum_{j\in V} x_{hj}^r = 0, \quad \forall h \in N, \quad \forall r \in R, \tag{4}$$

$$\sum_{i\in V} x_{oi}^r = 1, \quad \forall r \in R, \tag{5}$$

$$\sum_{i\in V} x_{io}^r = 1, \quad \forall r \in R, \tag{6}$$

$$\sum_{i\in N} q_i y_i^r \leqslant Q, \quad \forall r \in R, \tag{7}$$

$$t_i^r + s_i + t_{ij} - M(1 - x_{ij}^r) \leqslant t_j^r, \quad \forall (i,j) \in A, \quad \forall r \in R, \tag{8}$$

$$a_i y_i^r \leqslant t_i^r \leqslant b_i y_i^r, \quad \forall i \in N, \quad \forall r \in R, \tag{9}$$

$$t_o^r \geqslant \beta\sum_{i\in N} s_i y_i^r, \quad \forall r \in R, \tag{10}$$

$$t_i^r \leqslant t_o^r + t_{max}, \quad \forall i \in N, \quad \forall r \in R, \tag{11}$$

$$t_o^s + M(1 - z_{rs}) \geqslant t_o^{r'} + \beta\sum_{i\in N} s_i y_i^s, \quad \forall r,s \in R,\ r < s, \tag{12}$$

$$\sum_{r\in R}\sum_{s\in R|r<s} z_{rs} \geqslant |R| - K, \tag{13}$$

$$x_{ij}^r \in \{0,1\}, \quad \forall (i,j) \in A, \quad \forall r \in R, \tag{14}$$

$$y_i^r \in \{0,1\}, \quad \forall i \in N, \quad \forall r \in R, \tag{15}$$

$$z_{rs} \in \{0,1\}, \quad \forall r,s \in R,\ r < s, \tag{16}$$

$$t_i^r \geqslant 0, \quad \forall i \in N,\ r \in R. \tag{17}$$

The objective function (1) translates the fact that it is always desirable to visit as many customers as possible. Note that for the model to be valid, constant α has to be set to a value that ensures that. Constraints (4)–(6) are flow conservation constraints, and (7) and (13) define the vehicles' capacity and the size of the fleet, respectively. The fact that the visits to customers must respect their time windows is expressed in (9). Every two clients with consecutive visits in a same route must have compatible visit times (8), the same happening with two consecutive routes performed by a same vehicle (12). Finally, the setup time for every route must always be considered (10), (12).

2.3. A column generation model

In [3], the authors propose a branch-and-price algorithm to solve the MVRPTW. The master problem of the column generation scheme is a set covering model, where each column represents a vehicle's workday, w. A workday is a sequence of routes assigned to a vehicle, to be performed during the stipulated planning period.

Let Ω be the set of feasible workdays, d_w and g_w the total traveled distance and revenue of workday $w \in \Omega$, respectively, and a_{iw} a binary coefficient that indicates whether or not customer i is served in workday w. Binary variables x_w define if workday $w \in \Omega$ belongs to the solution, and the formulation of [3] states as follows.

$$\min \sum_{w\in\Omega}(d_w - \alpha g_w)x_w \tag{18}$$

$$\text{s.t.} \sum_{w\in\Omega} a_{iw}x_w \leqslant 1, \quad i \in V, \tag{19}$$

$$\sum_{w\in\Omega} x_w \leqslant K, \tag{20}$$

$$x_w \in \{0,1\}, \quad w \in \Omega. \tag{21}$$

The objective function (18) must ensure that a customer may only not be visited if it is not possible due to time or vehicle constraints. Parameter α must be set to a value that guarantees that. Constraints (19) state that each customer is visited at most once. The number of workdays in a solution can never exceed the number of available vehicles in the fleet (20).

The pricing problem generates feasible workdays, and it is formulated as an elementary shortest path problem with resource constraints, defined in a graph whose nodes represent vehicle routes, and whose arcs represent pairs of consecutive routes. This means that all feasible routes must be generated a priori.

3. A new pseudo-polynomial network flow model

In this section, we present a new network flow model for the MVRPTW, whose variables represent feasible vehicle routes. As in [3], all vehicle routes are previously generated. The integer model is then solved with a commercial software (CPLEX), explicitly considering all its variables. Because the nodes of the graph represent time instants, a discretization of time is required. The discretization will be discussed in the next section.

3.1. Vehicle routes generation

A route $r \in R$ may remain feasible when it begins at different time instants. Therefore, for every route r, we consider that there are several routes r_t, one for each possible departure instant t. The duration of a route r, σ_r, may be different for different departure instants, as the waiting times to serve customers may vary.

Let $(i_1,\ldots,i_{|N_r|})$ be the sequence of customers visited in route $r \in R$. The first possible time instant to end route r is $T'^{-}_r = \theta^r_{i_{|N_r|}} + s_{i_{|N_r|}} + t_{i_{|N_r|}o}$, being $\theta^r_{i_{|N_r|}}$ the first possible instant to start service at last customer $i_{|N_r|}$ in route r. It is possible to recursively calculate T'^{-}_r, considering that $\theta^r_{i_h} = \max\{\theta^r_{i_{h-1}} + s_{i_{h-1}} + t_{i_{h-1}i_h}, a_{i_h}\}$ for $h \in \{1,\ldots,|N_r|\}$ with $\theta^r_{i_0} = a_o$. As illustrated in Fig. 1, this means that beginning route r at any instant $t^*_r \leqslant T^-_r = \theta^r_{i_1} - t_{oi_1}$ implies ending it at instant T'^{-}_r. Therefore, it is clear that such a route is dominated by the route r that begins at instant T^-_r, and thus it may not be considered.

Similarly, the last possible time instant to end route r is $T'^{+}_r = \phi^r_{i_{|N_r|}} + s_{i_{|N_r|}} + t_{i_{|N_r|}o}$, being $\phi^r_{i_{|N_r|}}$ the last possible instant to start service at customer $i_{|N_r|}$ in route r and $\phi^r_{i_h} = min\{\phi^r_{i_{h-1}} + s_{i_{h-1}} + t_{i_{h-1}i_h}, b_{i_h}\}$, for $h \in \{1,\ldots,|N_r|\}$ with $\phi^r_{i_0} = b_o$. This means that beginning a route in any instant after $T^+_r = \phi^r_{i_1} - t_{oi_1}$ implies that the route is not feasible, as it does not respect at least one of the customers' time windows.

Note that if route r begins within the time interval $[T^-_r, T^+_r]$, it will have the minimum duration, as the waiting times are minimized.

For each $r \in R$, the interval $[T^-_r, T^+_r]$ is computed as described. The number of different feasible routes we consider is, therefore, equal to $\sum_{r\in R}\left\lceil \frac{T^+_r - T^-_r + 1}{U} \right\rceil$, being U the time unit. We consider $U = 1$.

3.2. A flow model for the MVRPTW

In our network flow model, every workday corresponds to a path in an acyclic directed graph $\Pi = (\Delta, \Psi)$. Its set of vertices $\Delta = \{0,1,\ldots,W\}$ represents discrete time instants from 0 to the workday length W and $\Psi = \{(u,v)^r : 0 \leqslant u < v \leqslant W, u \in [T^-_r, T^+_r], v = u + \sigma_r, r \in R\} \bigcup \{(u,v)^o : 0 \leqslant u < v \leqslant W, v = u + 1\}$ represents its set of arcs. Arcs correspond either to feasible vehicle routes or to waiting time periods (unit arcs). These waiting time arcs represent the instants of time, in a workday, that are spent by the vehicle at the depot. Note that in this model, we adjust the beginning time instant of each route $r \in R$ to $\beta\sum_{i\in N_r} s_i$ time instants before, in order to consider the loading time of the vehicle.

The model is formulated as a minimum flow problem. The number of constraints is polynomial in the size of W and the number of variables is polynomial in the size of W and the number of feasible routes, which is limited by a constant that depends on parameter

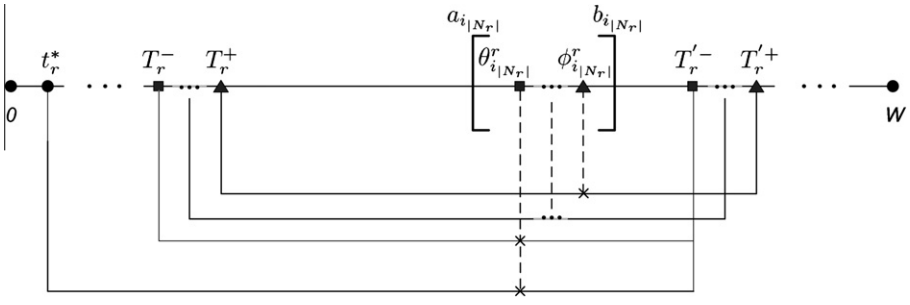

Fig. 1. First and last beginning and ending instants of route r.

t_{max}. Therefore, the model has a pseudo-polynomial number of variables and constraints.

Its variables λ^r_{uv} correspond to the flow in arc $(u,v)^r$, i.e., to the number of vehicles that go through route r, leaving the depot at instant u and arriving at instant v of their workday. Variable z corresponds to the total flow through the graph, and can be seen as the return flow from vertex W to vertex 0. Coefficient d_r represents the cost of route r, i.e., the sum of the total traveled distance in r. The model states as follows.

$$\min \sum_{(u,v)^r \in \Psi} \left(d_r - \alpha \sum_{i \in N_r} g_i \right) \lambda^r_{uv} \tag{22}$$

$$\text{s.t.} \sum_{(u,v)^r \in \Psi | i \in N_r} \lambda^r_{uv} \leqslant 1, \quad \forall i \in N, \tag{23}$$

$$-\sum_{(u,v)^r \in \Psi} \lambda^r_{uv} + \sum_{(v,y)^s \in \Psi} \lambda^s_{vy} = \begin{cases} z, & \text{if } v = 0, \\ 0, & \text{if } v = 1, \ldots, W-1, \\ -z, & \text{if } v = W, \end{cases} \tag{24}$$

$$z \leqslant K, \tag{25}$$

$$\lambda^r_{uv} \geqslant 0 \text{ and integer}, \quad \forall (u,v)^r \in \Psi, \tag{26}$$

$$z \geqslant 0 \text{ and integer}. \tag{27}$$

The objective is to minimize the total traveled distance of all vehicles during one workday (22). It may not be possible to visit all customers due to the limited number of available vehicles (25). The inequalities in constraints (23) express that. It is, however, always desirable to visit as many customers as possible.

Constraints (24) are the flow conservation constraints of the network. They ensure that the amount of flow that goes into a node is equal to the amount of flow that goes out of it.

The following example illustrates the structure of our model.

Example 3.1. Consider an instance of MVRPTW with five customers ($n = 5$), two available vehicles ($K = 2$) with a capacity of $Q = 10$ units, $t_{max} = 5$ and $\beta = 0.2$. Table 1 describes the coordinates (x_i, y_i), time window $[a_i, b_i]$, demand q_i and service time s_i for node $i \in N = \{1, \ldots, 5\} \bigcup \{o\}$. The distance between two nodes i and j is equal to the Euclidian distance between them. Table 2 lists all feasible routes, their beginning intervals and durations, as described in Section 3.1, and all arcs to consider in the model. Fig. 2 represents the network flow graph generated for this instance, and the corresponding optimal solution is shown in Fig. 3. In this solution, $z = 2$ and thus two vehicles are required, Λ_1 and Λ_2. By looking at the solution graph (Fig. 3), we can say that each of the vehicles has to perform two routes in its workday. One of the vehicles, Λ_1, would perform routes b and e, visiting customers 4, 2 and 3. It would start loading to perform route b at time instant 3.44, arrive at the depot at time instant 12.16, wait for 0.04 time instants, start loading for route e at time instant 12.20, arriving at the depot at time instant 21.65, where it would remain until the end of the workday. The second vehicle, Λ_2, would perform routes f and a, visiting customers 1 and 5. It would start loading to perform route f at time instant 3.60, arrive at the depot at time instant 8.00, wait for 2.99 time instants, start loading for route a at time instant 10.99, arriving at the depot at time instant 20.60, where it would remain until the end of the workday. In this solution, all customers are visited.

Table 1
Instance description (Example 3.1).

Customer	Coordinates	Time windows	Demand	Service time
i	(x_i, y_i)	$[a_i, b_i]$	q_i	s_i
0	(0,0)	[0,25]	0	0
1	(1,0)	[5,6]	1	2
2	(0,1)	[12,15]	7	2
3	(1,2)	[15,18]	1	2
4	(3,1)	[7,9]	2	2
5	(2,3)	[10,15]	3	2

3.3. Reductions based on dominance rules

The variables represent feasible vehicle routes in our network flow model. Clearly, reducing the number of arcs reduces the size of the model, increasing its efficiency. The next four propositions define dominance rules that allow us to discard some routes that are never interesting when compared to some other one, and also to reduce the number of different beginning instants for some of the routes.

The first dominance rule states that if two routes serve the same customers, a route can be dropped if it has a larger or equal cost and if it does not begin after nor end before the other.

Proposition 1. *Let $r, s \in R$ be two vehicle routes such that $N_r = N_s$. Let $t \in [T^-_r, T^+_r]$ and $t' \in [T^-_s, T^+_s]$. If $d_r \geqslant d_s$, $t \leqslant t'$ and $t + \sigma_r \geqslant t' + \sigma_s$, route $s_{t'}$ dominates route r_t.*

Proof. It is never better to consider route r_t over route $s_{t'}$, as the first one does not begin after nor end before the second one, both routes visit the same set of customers and the cost of r is not smaller than the cost of s. □

The second dominance procedure states that if all replications of a given route r end after the beginning of any other route, then the latest replication of r dominates the other replications of r.

Proposition 2. *Let $r \in R$ be a route such that $T^-_r + \sigma_r \geqslant max\{T^+_s : s \in R, s \neq r\}$. All routes $r_t, t \in [T^-_r, T^+_r[$, are dominated by route $r_{T^+_r}$.*

Proof. Let $T = max\{T^+_s : s \in R\}$. No route can begin after time instant T and route r will always end after time instant T. Consequently, if route r belongs to a vehicle's workday, it is always the last one. Therefore, any route $r_t, t \in [T^-_r, T^+_r[$, can be replaced by the route $r_{T^+_r}$ without losing the optimal solution, as $t \leqslant T^+_r$ and no other route will be performed after r. □

Table 2
Feasible routes (Example 3.1).

Route	Customers	Beginning interval	Duration	Arcs
r		$[T^-_r, T^+_r]$	σ_r	$\{(u,v)^r\}$
a	(5)	[5.99, 10.99]	9.61	$\{(6,15)^a; (7,16)^a; (8,17)^a; (9,18)^a; (10,19)^a; (11,20)^a\}$
b	(4)	[3.44, 5.44]	8.72	$\{(4,12)^b; (5,13)^b; (6,14)^b\}$
c	(3)	[12.36, 15.36]	6.88	$\{(13,19)^c; (14,20)^c; (15,21)^c; (16,22)^c\}$
d	(2)	[10.60, 13.60]	4.40	$\{(11,15)^d; (12,16)^d; (13,17)^d; (14,18)^d\}$
e	(2,3)	[10.20, 12.79]	9.45	$\{(11,19)^e; (12,20)^e; (13,21)^e\}$
f	(1)	[3.60, 4.60]	4.40	$\{(4,8)^f; (5,9)^f\}$

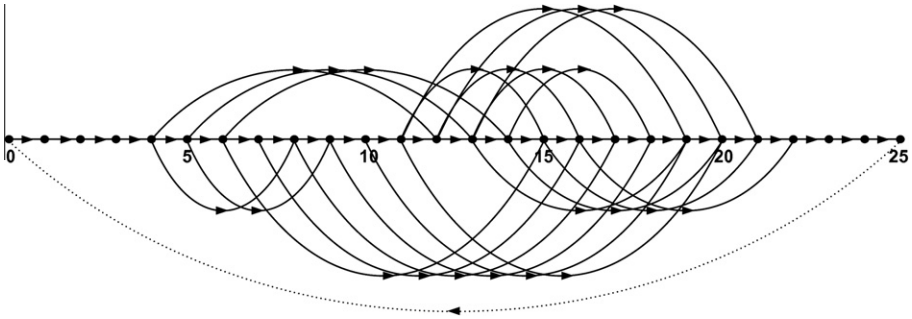

Fig. 2. Complete network flow graph (Example 3.1).

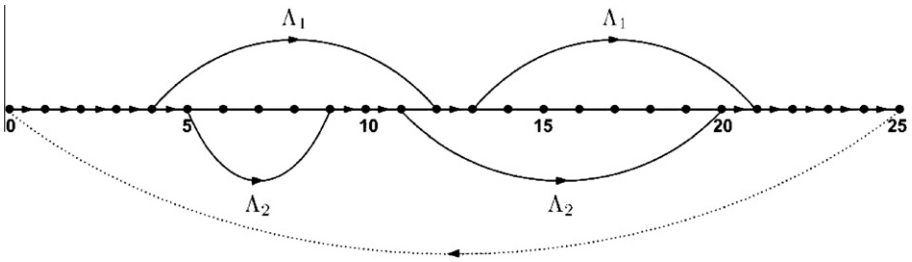

Fig. 3. Optimal solution (Example 3.1).

The third dominance rule states that if no routes begin in the interval of possible ends for a route s, then the latest replication of s dominates the other replications of s.

Proposition 3. *If* $\bigcup_{r\in R}[T_r^-, T_r^+] \bigcap [T_s^- + \sigma_s, T_s^+ + \sigma_s] = \varnothing$, *route* $s_{T_s^+}$ *dominates all routes* s_t, *with* $t \in [T_s^-, T_s^+[$.

Proof. Let (λ^*,z^*) be an optimal solution for the flow model of an instance of MVRPTW, where the route s_{t^-} (respectively s_{t^+}) precedes (resp. follows) the route s_t, in the same workday. This means that we have $\lambda^{*S^-}_{t^-,t'-} = \lambda^{*S}_{t,t'} = \lambda^{*S^+}_{t^+,t'+} = 1$. Since the solution (λ^*,z^*) is feasible, $T_s^+ \geqslant t \geqslant t'-$. If $\bigcup_{r\in R}[T_r^-, T_r^+] \bigcap [T_s^- + \sigma_s, T_s^+ + \sigma_s] = \varnothing$, we have that $t^+ > T_s^+ + \sigma_s$. Therefore, a solution (λ',z^*), where λ' equals λ^* by changing only the two components $\lambda'^S_{t,t'} = 0$ and $\lambda'_{T_s^+,T_s^+ + \sigma_s} s = 1$, remains feasible and has the same cost as (λ^*,z^*), and thus is an optimal solution. □

The fourth dominance rule is similar to the third. It states that if no routes end in the interval of possible beginning of a route s, then the earliest replication of s dominates the other replications of s.

Proposition 4. *If* $\bigcup_{r\in R}[T_r^- + \sigma_r, T_r^+ + \sigma_r] \bigcap [T_s^-, T_s^+] = \varnothing$, *route* $s_{T_s^-}$ *dominates all routes* $s_t, t \in]T_s^-, T_s^+]$.

Proof. Let (λ^*,z^*) be an optimal solution for the flow model of an instance of MVRPTW, where the route s_{t^-} (respectively s_{t^+}) precedes (resp. follows) the route s_t, in the same workday. This means that we have $\lambda^{*S^-}_{t^-,t'-} = \lambda^{*S}_{t,t'} = \lambda^{*S^+}_{t^+,t'+} = 1$. Since the solution (λ^*,z^*) is feasible, $T_s^- + \sigma_s \leqslant t' \leqslant t^+$. If $\bigcup_{r\in R}[T_r^- + \sigma_r, T_r^+ + \sigma_r] \bigcap [T_s^-, T_s^+] = \varnothing$, we have that $t'- < T_s^-$. Therefore, a solution (λ',z^*), where λ' equal to λ^* by changing only the two components $\lambda'^S_{t,t'} = 0$ and $\lambda'^S_{T_s^-,T_s^- + \sigma_s} = 1$, remains feasible and has the same cost as (λ^*,z^*), and thus is an optimal solution. □

Example 3.2. Consider an instance of the MVRPTW for which eight feasible vehicle routes, a, b, c, d, e, f, g and h, can be generated. The first graph in Fig. 4 represents the complete graph with all the replications of those eight routes, as described in Section 3.1. In the second graph, there are only the routes that need to be considered, after the arc reduction criteria described in Section 3.3 are applied.

T sets represent the beginning time instants of the routes and T' the ending time instants. Set T_b was reduced to its first point as it does not intersect any ending set (Proposition 4). Sets T_d and T_e were reduced to their last points, as all points in T'_d and T'_e are greater than T (Proposition 2). Finally, T_g was also reduced to its last point, as set T'_g does not intersect any beginning set (Proposition 3).

4. Convergent iterative relaxation based on the discretization

Our approach to solve an instance of a MVRPTW problem consists of enumerating all feasible routes, as described in Sections 3.1 and 3.3, and solving the corresponding network flow model.

The nodes of model (22)–(27) represent time instants. The distances (and thus the time) in the benchmarks we used are not integer, and thus we have two alternatives. Either we use a finely grained discretization (for example, each time unit would be 0.01), or we use some rounding procedures to use an integer time unit. The former alternative would lead to a network flow model with a huge number of variables and constraints, and would not allow a fast solution. Consequently, we used the latter alternative.

We discuss in the following the different ways of rounding the values. We chose a rounding strategy which slightly relaxes the problem. In many cases, the solution found remains feasible. However, it may happen that the solution found by model (22)–(27) is not feasible. Since our solution approach intends to be exact, we developed an algorithm that iteratively refines the discretization. It allows us to achieve the optimal value, despite our initial coarse discretization of time.

4.1. Initial rounding strategy

Recall that an arc $(u,v)^r$ in model (22)–(27) is related to a route r beginning at time u and ending at time v. Given that the set of vertices of graph Π is defined as a discrete set of values $\Delta = \{0,\ldots,W\}$,

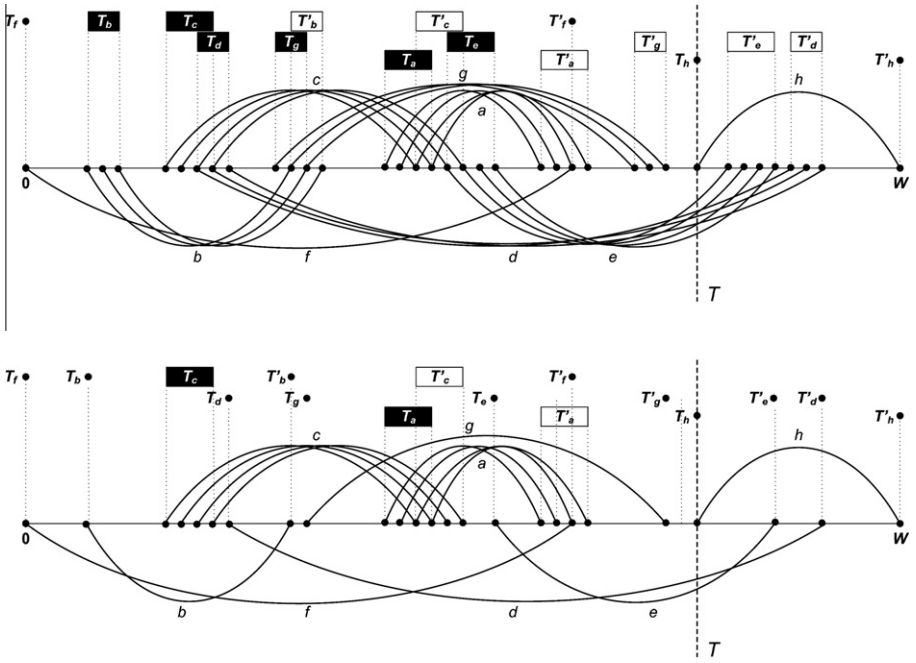

Fig. 4. Arcs reduction (Example 3.2).

it might be necessary for some arcs $(u,v)^r \in \Psi$, to round u and v to a value that belongs to set Δ. As mentioned before, we initially consider time units equal to 1, i.e., $\Delta = \{0,1,2,\ldots,W-1,W\}$. Several rounding procedures are possible, which are commented below.

- $u = \lfloor u \rfloor$ and $v = \lceil v \rceil$. In this case, in the model, a route begins slightly before and ends slightly after it actually does. It means that we may miss some solutions, but each feasible solution of (22)–(27) using this relaxation will be feasible for the initial problem. Consequently, this relaxation leads to a heuristic.
- $u = \lceil u \rceil$ and $v = \lfloor v \rfloor$. In this case, the model is slightly relaxed. It may happen that the solution found is not feasible. However it leads to a valid lower bound.
- ($u = \lceil u \rceil$ and $v = \lceil v \rceil$) or ($u = \lfloor u \rfloor$ and $v = \lfloor v \rfloor$). A lower bound is also obtained.

In our algorithm, we used the second rounding procedure: considering $u = \lceil u \rceil$ and $v = \lfloor v \rfloor$. We chose this technique for two reasons: the relaxation is expected to be tight, and infeasibilities are local and can be corrected, as we will explain below.

Note that infeasible solutions are only related to paths of the flow model (workdays) including two consecutive routes r and r' with one or less units of waiting time between them. For example, if a replication of route r ends at time 15.35, and a replication of route r' begins at time 15.15, our rounding procedure will make route r end at time 15 and route r' start at time 16, allowing r and r' to belong to the same working day even though this is not possible in the initial problem.

A first idea is to correct the solution by shifting route r backward or route r' forward to avoid this problem. If it makes the solution feasible, we prove the optimality, because the feasible solution will have the same set of routes and thus a cost equal to the lower bound. Practically speaking, after having obtained a solution x^*, we try to build a feasible solution, using the same routes found in solution x^*.

The algorithm works as follows. For each workday, we try to build a new path, only maintaining the sequence of routes. Let $(r_1,\ldots,r_p)$ be the sequence of routes in the workday, and T_{r_i} the new beginning and T'_{r_i} the new end of route r_i, $\forall i \in \{1,\ldots,p\}$. We set $T_{r_i} = max(T^-_{r_i}, T'_{r_{i-1}})$. If $T_{r_i} \leqslant T^+_{r_i}$, $\forall i \in 1,\ldots,p$, the solution is feasible. If not, we cannot prove feasibility, and another algorithm has to be used.

4.2. Iteratively refining the discretization

Our solution approach consists in iteratively fixing the infeasibilities due to discretization issues. At each step of the algorithm, we detect all the time instants where infeasibilities occur. For each of these time instants, we locally modify the discretization, adding the fractional values needed to refine the relaxation entailed by the initial discretization.

The initial relaxation can be seen as an aggregation of many time instants into a unique integer one. Our refinement technique is equivalent to a *disaggregation* of some nodes of the current graph. For each pair of conflicting arcs $(u,v)^r$ and $(u',v')^{r'}$, we consider the fractional values of v and u' in order to correct the solution. Our solution approach is summarized in Algorithm 1.

Algorithm 1: Iterative Disaggregation Algorithm

```
Input: Instance I of the MVRPTW
output: Optimal solution x*
optimal = False;
While optimal = False do
    Solve I with model(22)–(27), possibly obtaining an
     infeasible solution x*
    Check if x* is feasible;
    if x* is feasible then
        optimal = True;
    else
        Apply node disaggregation;
```

The node disaggregation process works as follows. We check all the arcs that belong to the solution. If there are at least two conflicting arcs $(u,P)^r$, $(P,v)^s$ in the solution, we decompose the integer

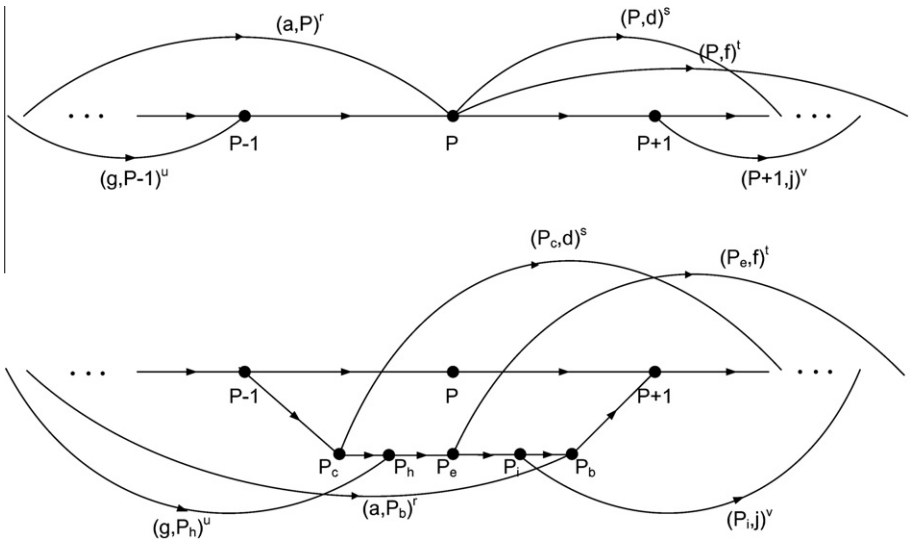

Fig. 5. Node disaggregation (Example 4.1).

node P. In order to do so, we check all arcs of the model $(w,z)^r \in \Psi$ such that $w = P$ or $z = P$ (except the waiting arcs), setting them to their original values $w' = P - 1 + \xi_a$ or $z' = P + \xi_b$, with $0 \leqslant \xi_a,\ \xi_b < 1$. We also check for all arcs $(x,y)^r \in \Psi$ such that $y = P - 1$ or $x = P + 1$, setting them to their original values $y' = P - 1 + \xi_c$ or $x' = P + \xi_d$, with $0 \leqslant \xi_c,\ \xi_d < 1$. When all the new η values are calculated, we sort them by their increasing values, and add the new nodes P_q, $q \in \{1,\ldots,\eta\}$, to the graph, always respecting their order.

Example 4.1. Consider an instance of the MVRPTW with an optimal solution with three incident arcs in node P, $(a,b)^r$, $(c,d)^s$, $(e,f)^t$ with $b = c = e = P$, an arc $(g,h)^u$ with $h = P - 1$ and an arc $(i,j)^v$ with $i = P + 1$. Let P_b, P_c, P_e, P_h and P_i be the original non rounded values of b, c, e, h and i respectively. Fig. 5 illustrates the graph transformation.

We also need to check if there are pairs of arcs in the solution, $(u,v)^r$ and $(x,z)^s$, such that $v = P$ and $x = P + 1$ and their original non rounded values $v' = P + \xi_v$ and $x' = P + \xi_x$ are such that $\xi_v > \xi_x$. In such cases, all beginning points incident in $P + 1$ or ending points incident in P must be set to their non rounded values.

Example 4.2. Consider an instance of the MVRPTW for which arcs $(a,b)^s$ and $(c,d)^t$, with $b = P$ and $c = P + 1$, belong to the optimal solution. All ending points incident in P or beginning points incident in $P + 1$ must be set to their original values. Fig. 6 illustrates the disaggregation procedure for this case.

5. Computational results

Our algorithm was tested and compared with the branch-and-price algorithm described in [3]. In order to do so, we used the same set of instances and the same values of parameters.

The comparisons are performed on the well known Solomon instances for the VRP with time windows [19] that were used by [3]. The instances are divided into three different categories, *RC* with 8 instances, *R* with 11 instances and *C* with 8 instances, according to the distribution of the customers' locations. Each category is

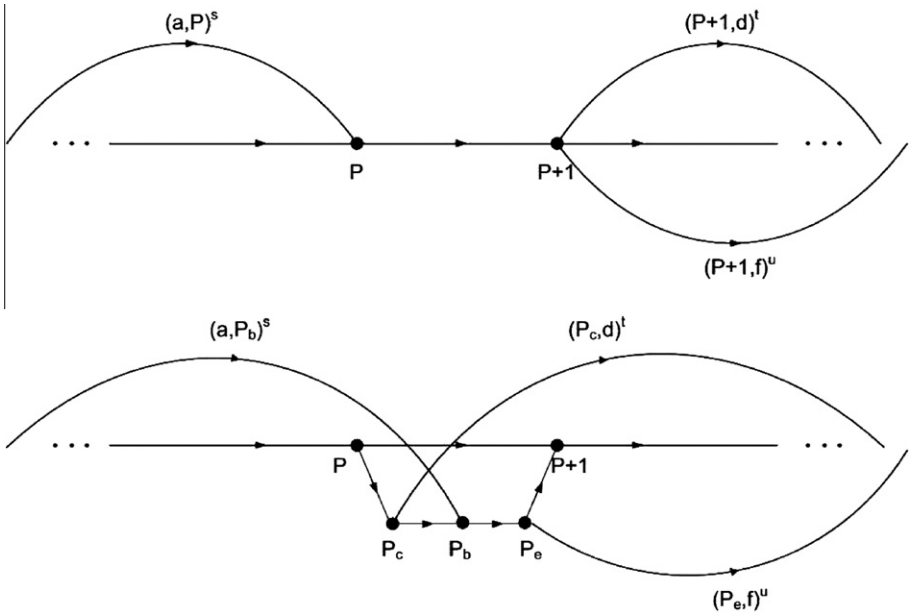

Fig. 6. Node disaggregation (Example 4.2).

divided into two groups (1 and 2). We are only considering the second group instances, as the first group ones have small time window, which would compromise the existence of workdays with more than one route. Originally, each instance has 100 customers but we only consider the n first ones, similarly to [3]. The value n is specified for each set of computational results.

The algorithm was implemented in C++ and the network flow model was solved with ILOG CPLEX 12.1. Note that the computational tests were run on a PC with a 2.66 GHz Quad-Core processor and 4 GB of RAM, whereas the tests in [3] were run on an AMD Opteron 3.1 GHz with 16 GB of RAM. Clearly, our machine is less powerful than the machine of these authors. In fact, the performance ratio between these two processors is at most equal to 0.94. This factor may be used as an estimation to convert our computing times for the machine used in [3].

The tests were run for $K = 2$, $\alpha = 2max_{i,j\in N}d_{ij} + 1$, $\beta = 0.2$, $g_i = 1$, $\forall i \in N$ and $t_{ij} = d_{ij}$, $\forall i, j \in V$.

All instances were run considering the first 25 and 40 clients. Each of these instances was tested for two different values of t_{max}. First for a smaller one (75 for instances of groups *RC* and *R*, and 220 for group *C*) which results in less routes to consider, making the problem easier to solve, and then for a larger one (100 for instances of groups *RC* and *R*, and 250 for group *C*).

Table 3 shows the impact of the arc reduction described in Section 3.3. Each line represents the average values of a group of instances, belonging to one of the three different categories, *RC*, *R*, or *C*, with n clients and a given t_{max}. The other columns represent the following: $|R|$ is the number of different routes and $|\Upsilon|$ is the number of all routes for all their beginning instants, as described in Section 3.1; $|R^{red}|$ and $|\Upsilon^{red}|$ represent these same values, after the reduction procedure has been applied; finally, $Dif_R\% = \frac{|R|-|R^{red}|}{|R|}\%$ and $Dif_\Upsilon\% = \frac{|\Upsilon|-|\Upsilon^{red}|}{|\Upsilon|}\%$.

We were able to reduce approximately 29% of the number of variables to include in the model.

We now report the results obtained by our algorithm on the instances used by [3]. For these results, the reduction procedures are applied before running the procedure. Tables 4–7 describe the computational results for the instances that were solved to optimality by at least one of the two methods tested. Several combinations of the number of customers and values t_{max} are tested. We ran the instances for a maximum time of 7200 s. The number of iterations of the iterative disaggregation algorithm is represented by n_{it} and z_{IDA} stands for the optimal value obtained with our approach. Columns t_{IDA} and t_{AGP} show the time, in seconds, required to solve the iterative disaggregation algorithm and the branch-and-price algorithm in [3], respectively. The percentage of visited customers is represented by *% Cust*, and the last column t_{Red} represents $[(t_{AGP} - t_{IDA})/t_{AGP}]\%$.

We solved around 62% of all instances, whereas in [3] only about 37% of the instances were solved to optimality. We were able to solve 28 instances that were not solved by [3]. On the contrary, one instance is solved by this latter method and not by ours. However, note that in [3] the authors need much more than 7200 seconds to solve this instance. The difference in the number of instances solved is larger for the instances with $n = 40$ and with smaller t_{max}. For this set of instances, we were able to solve 16 instances, whereas the method of [3] only solved 7 instances.

If we consider the instances solved by both methods, ours solved them in approximately 2% of the time reported in [3]. Our method is faster for every case with a reduction of at least 91%, except for two instances where the reduction is smaller, and the notable exception of two other instances for which our method is slower than the method of [3]. These two instances involve 40 customers and a $t_{\max}$ equal to 75. In one of these two instances, the number of disaggregation iterations is 4, and in the other, the model ran three times.

In what concerns the number of iterations of the algorithm, only in six cases out of the sixty-seven solved was it necessary to run the model more than once. The model was solved twice and thrice in one case each and four times for four test cases. This means that our algorithm always converged in four or less iterations.

As expected, when the number of customers increases it becomes harder to solve the problem to optimality. This also happens

Table 4
Computational results for 25 customers and t_{max} 75 and 220.

Inst	n	t_{max}	n_{it}	*%Cust*	z_{IDA}	t_{IDA}	t_{AGP}	t_{Red} (%)
RC201	25	75	1	100	988.2	0.3	3.1	91.29
RC202	25	75	1	100	881.6	37.2		
RC203	25	75	1	100	749.26	54.2		
RC204	25	75	1	100	744.83	171.0		
RC205	25	75	1	100	840.47	1.6	28.8	94.48
RC206	25	75	1	100	761.14	2.0	7156.8	99.97
R201	25	75	1	100	762.53	0.5	68.3	99.22
R202	25	75	1	100	645.86	3.1	205.2	98.51
R203	25	75	1	100	622.04	10.6	1333.2	99.21
R204	25	75	1	100	579.75	106.2	30983.3	99.66
R205	25	75	1	100	634.17	1.5	354.1	99.59
R206	25	75	1	100	596.81	4.7	318.4	98.52
R207	25	75	1	100	585.81	19.4	2853.5	99.32
R208	25	75	1	100	579.75	66.0	9270.3	99.29
R209	25	75	1	100	602.47	4.9	262.6	98.12
R210	25	75	1	100	636.24	11.8	5094.1	99.77
R211	25	75	1	100	575.97	64.5	5648.6	98.86
C201	25	220	1	100	659.15	10.6	40361.2	99.97
C202	25	220	1	100	653.5	212.4		
C203	25	220	1	100	646.51	233.9		
C204	25	220	1	100	602.58	423.0		
C205	25	220	1	100	636.52	34.7		
C206	25	220	1	100	636.52	40.2		
C207	25	220	1	100	603.34	29.5		
C208	25	220	1	100	613.34	12.9		
Solved$_{AF}$:		92.59%	*Solved*$_{AGP}$:		55.56%	*Average* t_{Red}:		98.39%

Table 3
Arc reduction.

Inst	n	t_{max}	$\|R\|$	$\|\Upsilon\|$	$\|R^{red}\|$	$\|\Upsilon^{red}\|$	$Dif_R\%$	$Dif_\Upsilon\%$
RC	25	75	516.75	116631.13	419.13	71673.63	18.90	34.13
R	25	75	702.64	216311.18	618.82	149259.09	12.64	28.89
C	25	220	336.00	183614.25	318.86	128034.00	16.55	28.58
RC	40	75	641.13	154921.25	520.75	96329.75	19.07	33.44
R	40	75	3661.82	1247141.73	3266.73	899976.00	11.36	25.62
C	40	220	1530.63	1153261.75	804.57	313353.29	16.83	26.40
RC	25	100	4463.88	746806.00	3374.88	411120.38	21.48	37.63
R	25	100	4705.91	1226177.91	4154.18	816231.18	12.15	29.98
C	25	250	1308.25	610079.88	1237.29	452880.29	16.34	26.84
RC	40	100	5648.88	1096099.38	4357.13	614915.63	21.51	37.41
R	40	100	5995.50	424145.00	5057.50	342092.50	14.38	16.80
C	40	250	2891.00	835131.67	2291.83	576158.67	16.92	22.53
Average			2700.20	667526.76	2201.80	406002.03	16.51	29.02

Table 5
Computational results for 40 customers and t_{max} 75 and 220.

Inst	*n*	t_{max}	n_{it}	*%Cust*	z_{IDA}	t_{IDA}	t_{AGP}	t_{Red} (%)
RC201	40	75	4	77.50	1292.35	29.4	14.6	−50.37
RC202	40	75	1	92.50	1458.09	40.2	6823.2	99.41
RC205	40	75	3	85	1290.75	6992.6	1904.2	−72.77
R201	40	75	1	95	1130.73	2358.8	2979.5	20.83
R203	40	75	1	100	962.42	436.0		
R205	40	75	4	100	1019.89	3263.7	244494.0	98.67
R206	40	75	1	100	931.94	209.9		
R209	40	75	1	100	935.95	771.3		
R210	40	75	4	100	963.45	1803.9		
C201	40	220	1	100	1169.04	25.5	19978.9	99.87
C202	40	220	1	100	1111.34	79.4		
C203	40	220	1	100	1089.24	342.3		
C205	40	220	1	100	1084.02	63.6		
C206	40	220	1	100	1081.57	109.3		
C207	40	220	1	100	1055.24	659.0		
C208	40	220	1	100	1072.22	112.7	3221.7	96.5
$Solved_{AF}$:		59.26%	$Solved_{AGP}$:		25.93%	*Average* t_{Red}:		41.73%

Table 6
Computational results for 25 customers and t_{max} 100 and 250.

Inst	*n*	t_{max}	n_{it}	*%Cust*	z_{IDA}	t_{IDA}	t_{AGP}	t_{Red} (%)
RC201	25	100	1	100	849.45	2.0	46.3	95.7
RC202	25	100	1	100	679.95	11.6	1096.3	98.94
RC203	25	100	1	100	593.63	47.0		
RC205	25	100	1	100	702.61	8.2	262.8	96.86
RC206	25	100	1	100	604.23	8.0	222.7	96.42
RC207	25	100	2	100	514.9	91.7		
R201	25	100	1	100	698.26	1.3	43.6	96.95
R202	25	100	1	100	617.6	32.6	25249.9	99.87
R203	25	100	1	100	577.8	64.1	75729.3	99.92
R205	25	100	1	100	559.21	9.4	1202.3	99.22
R206	25	100	1	100	523.7	40.0	28498.1	99.86
R209	25	100	1	100	517.74	47.7	11173.9	99.57
R210	25	100	1	100	547.29	58.9	26690.2	99.78
C201	25	250	1	100	541.02	0.4	1.3	68.46
C202	25	250	1	100	533.55	167.9		
C205	25	250	1	100	530.05	3.7	116.6	96.87
C206	25	250	1	100	527.95	20.7	1987.2	98.96
C207	25	250	1	100	525.57	44.2		
C208	25	250	1	100	525.57	63.2		
$Solved_{AF}$:		70.37%	$Solved_{AGP}$:		51.85%	*Average* t_{Red}:		96.2%

Table 7
Computational results for 40 customers and t_{max} 100 and 250.

Inst	*n*	t_{max}	n_{it}	*%Cust*	z_{IDA}	t_{IDA}	t_{AGP}	t_{Red} (%)
RC201	40	100	1	85	1157.65	3.6	77.8	95.36
RC202	40	100	4	97.5	1322.08	1013.8		
RC205	40	100	1	95	1195.51	35.7	4733.3	99.25
R201	40	100					127424.0	
C201	40	250	1	100	966.89	6.4	659.2	99.03
C205	40	250	1	100	921.37	88.5		
C206	40	250	1	100	919.24	290.5		
C208	40	250	1	100	915.61	491.5		
$Solved_{AF}$:		25.93%	$Solved_{AGP}$:		14.81%	*Average* t_{Red}:		97.88%

when we increase the value of t_{max}, as the bigger this parameter is, the larger is the number of feasible routes, and thus variables, to consider.

6. Conclusions

The MVRPTW is a variant of the classical vehicle routing problem that has received little attention in the literature. In this paper, we described a new network flow model, and an exact solution algorithm to solve this problem. We conducted computational experiments on a set of benchmark instances from the literature. Our approach proved to be much more efficient than other methods described in the literature.

One of the aspects that we will address as future work is related to the development of efficient schemes for managing the number of variables and constraints of the model. Indeed, the sometimes huge number of arcs that it may require remains an issue that prevents the resolution of larger instances. A key challenge will be to extend our approach so as to address efficiently these larger size instances. This may be accomplished by developing new hybrid algorithms or by trying to generate arcs dynamically, instead of explicitly considering them all from the beginning. Alternatively, the development of model-based heuristics that take advantage of the properties of this pseudo-polynomial model will also be explored.

Acknowledgements

This work was partially supported by the Portuguese Science and Technology Foundation through the doctoral grant SFRH/BD/39607/2007 for Rita Macedo, by the Algoritmi Research Center of the University of Minho for Cláudio Alves and José Valério de Carvalho, by the Université de Lille 1 and INRIA Lille Nord Europe for François Clautiaux, and by the International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the French Ministry of Higher Education and Research, and the French National Center for Scientific Research for Saïd Hanafi.

The authors thank the anonymous referees for their constructive comments, which helped improving the quality of the paper.

References

[1] F. Alonso, M.J. Alvarez, J.E. Beasley, A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions, Journal of the Operational Research Society 59 (2008) 963–976.
[2] N. Azi, M. Gendreau, J.-Y. Potvin, An exact algorithm for a single-vehicle routing problem with time windows and multiple routes, European Journal of Operational Research 178 (2007) 755–766.
[3] N. Azi, M. Gendreau, J.-Y. Potvin, An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles, European Journal of Operational Research 202 (2010) 756–763.
[4] J. Brandão, A. Mercer, The multi-trip vehicle routing problem, Journal of the Operational Research Society 49 (1998) 799–805.
[5] Olli Bräysy, Michel Gendreau, Vehicle routing problem with time windows, part I: Route construction and local search algorithms, Transportation Science 39 (1) (2005) 104–118.
[6] Olli Bräysy, Michel Gendreau, Vehicle routing problem with time windows, part II: Metaheuristics, Transportation Science 39 (1) (2005) 119–139.
[7] G. Clarke, J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research 12 (4) (1964) 568–581.
[8] J.-F. Cordeau, G. Laporte, M.W.P. Savelsbergh, D. Vigo, Vehicle Routing. Transportation, in: C. Barnhart, G. Laporte, (eds.), Handbooks in Operations Research and Management Science 14 (2007) 367–428.
[9] G.B. Dantzig, J.H. Ramser, The truck dispatching problem, Management Science 6 (1) (1959) 80–91.
[10] K. Doerner, G. Fuellerer, M. Gronalt, R. Hartl, M. Iori, Metaheuristics for the vehicle routing problem with loading constraints, Networks 49 (4) (2007) 294–307.
[11] B. Fleischmann, The vehicle routing problem with multiple use of vehicles, working paper Fachbereich Wirtschaftswissenschaften, Universität Hamburg, Germany, 1990.
[12] G. Fuellerer, K. Doerner, R. Hartl, M. Iori, Metaheuristics for vehicle routing problems with three-dimensional loading constraints, European Journal of Operational Research 201 (3) (2010) 751–759.
[13] M. Gendreau, M. Iori, G. Laporte, S. Martello, A Tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints, Networks 51 (4) (2008) 4–18.
[14] M. Iori, J.J. Salazar-González, D. Vigo, An exact approach for the vehicle routing problem with two-dimensional loading constraints, Transportation Science 41 (2) (2007) 253–264.

[15] A. Oliveira, O. Vieira, Adaptive memory programming for the vehicle routing problem with multiple trips, Computers & Operations Research 34 (2007) 28–47.
[16] R.J. Petch, S. Salhi, A multi-phase constructive heuristic for the vehicle routing problem with multiple trips, Discrete Applied Mathematics 133 (2004) 69–92.
[17] S. Salhi, R.J. Petch, A GA based heuristic for the vehicle routing problem with multiple trips, Journal of Mathematical Modelling and Algorithms 6 (2007) 591–613.
[18] A. Şen, K. Bülbül, A survey on multiple vehicle routing problem, in: VI International Logistics and Supply Chain Congress' 2008, November 6–7, 2008, Istambul, Turkiye.
[19] M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Operations Research 35 (2) (1987) 254–265.
[20] E. Taillard, G. Laporte, M. Gendreau, Vehicle routing with multiple use of vehicles, Journal of the Operational Research Society 47 (1996) 1065–1070.
[21] P. Toth, D. Vigo, The vehicle routing problem, SIAM Monographs on Discrete Mathematics and Applications 9 (2002).