Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 824961, 13 pages
http://dx.doi.org/10.1155/2013/824961

Research Article

Column Generation for a Multitrip Vehicle Routing Problem with Time Windows, Driver Work Hours, and Heterogeneous Fleet

Michel Povlovitsch Seixas and André Bergsten Mendes

Department of Naval Architecture and Ocean Engineering, University of São Paulo, Avenida Professor Mello Moraes 2231, 05508-030 São Paulo, SP, Brazil

Correspondence should be addressed to André Bergsten Mendes; andbergs@usp.br

Received 3 December 2012; Revised 29 January 2013; Accepted 4 February 2013

Academic Editor: Rubén Ruiz García

This study addresses a vehicle routing problem with time windows, accessibility restrictions on customers, and a fleet that is heterogeneous with regard to capacity and average speed. A vehicle can perform multiple routes per day, all starting and ending at a single depot, and it is assigned to a single driver whose total work hours are limited. A column generation algorithm is proposed. The column generation pricing subproblem requires a specific elementary shortest path problem with resource constraints algorithm to address the possibility for each vehicle performing multiple routes per day and to address the need to set the workday's start time within the planning horizon. A constructive heuristic and a metaheuristic based on tabu search are also developed to find good solutions.

1. Introduction

Transportation is an important area for logistics studies, and on average, it absorbs a higher percentage of the total logistics costs than other logistic activities [1]. Vehicle routing problems belong to a broad category of operational research problems known as network optimisation problems.

A variant of the vehicle routing problem with time windows (VRPTW) is considered here, with the highlights that each vehicle is assigned to a single driver who can travel several routes during a workday and be within the limit of his/her work hours. This feature requires the optimal point in time to start each route to be identified. Some customers have accessibility restrictions, thereby requiring specific vehicles. Concerning the objective function, the purpose is to minimise the total routing cost, which depends on both the distance travelled and vehicles used.

Great advancements have been obtained for different VRPTW variants, such as those by Solomon and Desrosiers [2] and Desrochers et al. [3]. Approaches with column generation for set partitioning formulations of the VRPTW and other variants have been presented as well. Desrochers et al. [4] proposed the use of the column generation technique to solve the linear relaxation of the VRPTW set partitioning formulation. Columns are added, when necessary, by means of the resolution of the shortest path problem with time windows and capacity constraints using a dynamic programming algorithm. The solutions obtained generally yielded excellent dual bounds which are used in a branch-and-bound algorithm to solve the integer formulation. This same approach is used in the present study.

Obtaining solutions to the vehicle routing problem with multiple routes, but without time windows, has been approached by means of heuristics by Taillard et al. [5], Brandão and Mercer [6], Olivera and Viera [7], and Petch and Salhi [8, 9], among others. In the study by Taillard et al. [5], different routes generated by tabu search are combined to produce workdays through the resolution of a

bin-packing problem. In the present work, the tabu search acts on a space with multiple routes without the need to combine them to form workdays. This approach, which is more costly for seeking neighbour solutions within a larger space, eliminates the computational cost of analysing the feasibility of combining different routes and finding a good combination.

Azi et al. [10] proposed a method, which is either exact or heuristic, to solve the VRPTW with multiple routes. It is a generalization of a previous method (Azi et al. [11]) for the same problem, but with only one vehicle available. In their work, a preprocessing is conducted to generate the set of all feasible routes or a subset of these routes so that a branch-and-price algorithm is employed to find the optimal solution over the set of routes generated during the preprocessing stage. In the branch-and-price algorithm of Azi et al. [10], the pricing subproblem, which can be cast as an elementary shortest path problem with resource constraints, consists of finding a sequence of feasible routes on a network where each vertex corresponds to such a route. Alternatively, in the present study, the subproblem consists of finding a sequence of customers on a network where a vertex corresponds to a customer or the depot, and no preprocessing is required to find the optimal solution over the set of all feasible routes.

Concerning the treatment typically given to a workday, it is considered that a number of working hours cannot be exceeded for any route, forcing the last customer to be served within a maximum time interval from when the route began [10, 12]. Cordeau and Laporte [13] proposed a tabu search heuristic for the static dial-a-ride problem with route duration constraints that evaluates the capacity to delay the departure time from the depot to help improve solution feasibility during the neighbourhood search. In the present problem, a similar control is necessary, but it is applied to multiple routes forming a workday in an exact, rather than heuristic, context.

Ceselli et al. [14] solved a real-world, multitrip vehicle routing problem with driver constraints that are similar to those in this study. In their work, each vehicle is described by a maximum daily duty length and, according to work rules, the waiting time and unloading time count as rest time or driving time. In the present problem, the waiting time must always be counted as driving time, which requires the ability to handle the freedom of a variable workday start time.

It is also worth stressing that classical dynamic programming algorithms developed to solve the shortest path problem with resource constraints, such as in Feillet et al. [15], do not include considerations about the determination of the beginning of each vehicle path in relation to the planning horizon. In the present study, the instant when the workday starts is a decision to be made together with the determination of the sequence of customers to be visited.

This characteristic, together with the possibility of the vehicle being able to carry out multiple routes per day, is treated by a specific algorithm to solve the subproblem in the ambit of column generation. This is the major contribution of this work, in addition to providing a framework that also contains a constructive heuristic and a very effective tabu search.

It should be stressed that the vehicle routing problem considered in this study generalizes the VRPTW and is therefore NP hard [16].

The paper is organised as follows: Section 2 provides a brief description of the characteristics of the problem studied and of the objective function to be optimised; Section 3 presents a mathematical formulation; Section 4 describes procedures and algorithms; Section 5 presents parameter settings, problem sets, and the results obtained. The conclusions are provided in Section 6.

2. Problem Description

The vehicle routing problem studied in this work consists of determining the workday of each vehicle in the fleet to minimise the total cost of the distribution operation from a single depot.

A vehicle workday is defined as a sequence of customers to be visited; for this purpose, the vehicle may return to the distribution centre to be reloaded and then start a new delivery route. That is, a workday route is defined as a sequence of customers who must be visited by the vehicle in a specified order after the vehicle is loaded at the distribution centre. Thus, a workday consists of a feasible sequence of routes throughout the day. For convenience purposes, the maximum number of routes in a workday is predetermined. Vehicles that are not used run a fictitious route in which a vehicle leaves the distribution centre and returns to it without visiting any customer at no cost.

The demand of each customer is known before the day starts, and every customer must be served only once by a vehicle within the planning horizon; that is, a delivery is not fragmented and is compulsory.

Each customer has a specific service time that depends on the vehicle used to perform the service and a strict time window that must be respected. If the vehicle reaches a customer before the opening of its time window, it must wait until the time window opens to begin the service. If the vehicle arrives after the end of the time window, the service is not provided.

Some customers can only be visited by specific vehicles due to accessibility restrictions at the delivery location.

The fleet is heterogeneous in terms of both capacity and average displacement speed. The cost of a vehicle is variable, depending only on the total distance travelled and vehicle type.

The total number of hours worked by a driver in a workday cannot exceed a specified maximum limit. The vehicle loading time at the distribution centre before the route begins and possible waiting times at customers' locations until the opening of their time windows are considered in the calculation of the worked hours, along with the travelling time between customers and the service time at each customer's location. The loading time of the vehicle at the distribution centre depends on the vehicle type.

The travelling times between customers and between the distribution centre and customers depend on both the distance travelled and the average speed of the vehicle used.

3. Mathematical Formulation

Let digraph $G^* = (N \cup \{O_1\} \cup \{O_2\}, A^*)$ be given where N is the set of customers; O_1 and O_2 are fictitious origin and destination vertices, respectively, both corresponding to the depot; A^* is the set of arcs joining two distinct vertices of G^*; and any feasible route corresponds to a path from O_1 to O_2 on G^*.

Let R represent the maximum number of routes driven during a vehicle workday. To represent a workday with R routes, digraph G^* is sequentially replicated R times, forming digraph G, in which a path from O_1 (origin of route 1) to O_{R+1} (destination of route R) represents a vehicle workday with R routes (see Figure 1).

Computationally, only a reduced digraph of a single route G^* is stored and used so that memory consumption is not significantly increased.

An instance of the problem is defined by the data listed below.

(i) Digraph $G = (\{N_1, N_2, \ldots, N_R\} \cup \{O_1, O_2, \ldots, O_R, O_{R+1}\}, A)$ in which the set of vertices is composed of the set of customers N replicated R times by adding the R route origins and the vertex of the final destination (O_{R+1}); set A contains the arcs between any two vertices on the same route. There are no arcs reaching origin O_1, and there are no arcs leaving destination O_{R+1}. Vehicles not used in route r during the workday travel the fictitious arc (O_r, O_{r+1}) at zero cost.

(ii) Set of heterogeneous vehicles K.

(iii) N^K—set of customers to be served by specific authorised vehicles: $N^K \subseteq N$.

(iv) Each customer $i \in N^K$ has a group of authorised vehicles $K_i : K_i \subseteq K$.

(v) Q_k—maximum capacity of vehicle k, in units.

(vi) $[a_i, b_i]$—time window of customer i. For the vertices $\{O_1, \ldots, O_{R+1}\}$ representing the distribution centre (depot), the time windows correspond to the planning horizon; that is, $[a_{O_1}, b_{O_1}] = \cdots = [a_{O_{R+1}}, b_{O_{R+1}}] = [a_{DC}, b_{DC}] = [0, b_{DC}]$.

(vii) D_i—demand of customer i, in units.

(viii) s_i^k—service time at customer i when the service is provided by vehicle k. At the distribution centre, this corresponds to the average loading time of vehicle k, s_{DC}^k, which is independent of the customers to be served along route, on the grounds that the pallets are ready to be loaded at the beginning of each route and also on the grounds that the vehicle type is more significant due to manoeuvring times, parking time, and need for equipment use such as forklifts.

(ix) c_{ijr}^k—cost of travelling the arc (i, j) in route r with vehicle $k \in K$. This value depends on the arc (i, j) distance, d_{ij}, and the variable cost of the vehicle k, VC_k (\$/unit of distance).

(x) $\tilde{t}_{ij}^k$—travel time of arc (i, j) with vehicle k.

(xi) $W_{\max,k}$—maximum driver working hours for vehicle k, that is, the maximum duration of a vehicle k workday.

(xii) R—maximum number of routes allowed per day to the vehicles. This value is an upper bound that is sufficiently large to hold all possible trips according to past daily routing operation experience.

(xiii) M_{ij}^k is a sufficiently large number defined as: $M_{ij}^k = \max\{b_i + s_i + \tilde{t}_{ij}^k - a_j, 0\}$, $k \in K$, $(i, j) \in A$.

The mathematical formulation has two decision variables: x_{ijr}^k, a binary variable that is equal to 1 if arc (i, j) is travelled by vehicle k along route r and that is equal to 0 otherwise, and t_{ir}^k, a variable that defines the instant in time at which vehicle k will serve customer i along route r.

Below is the mathematical formulation of the initial problem, denoted as IP

$$\text{Min} \sum_{r=1}^{R} \sum_{k \in K} \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijr}^k \tag{IP}$$

s.t.

$$\sum_{r=1}^{R} \sum_{k \in K} \sum_{j \in N \cup \{O_{r+1}\}} x_{ijr}^k = 1 \quad i \in N, \tag{1}$$

$$\sum_{r=1}^{R} \sum_{i \in N \cup \{O_r\}} x_{ijr}^k = 0 \quad j \in N^K,\ k \notin K_j, \tag{2}$$

$$\sum_{j \in N \cup \{O_{r+1}\}} x_{O_r jr}^k = 1 \quad k \in K,\ r = 1, \ldots, R, \tag{3}$$

$$\sum_{i \in N \cup \{O_r\}} x_{iO_{r+1}r}^k = 1 \quad k \in K,\ r = 1, \ldots, R, \tag{4}$$

$$\sum_{i \in N \cup \{O_r\}} x_{ihr}^k - \sum_{j \in N \cup \{O_{r+1}\}} x_{hjr}^k = 0 \quad k \in K,\ h \in N,\ r = 1, \ldots, R, \tag{5}$$

$$a_i \le t_{ir}^k \le b_i \quad k \in K,\ r = 1, \ldots, R,\ i \in N \cup \{O_r, O_{r+1}\}, \tag{6}$$

$$t_{ir}^k + s_i^k + \tilde{t}_{ij}^k - M_{ij}^k \left(1 - x_{ijr}^k\right) \le t_{jr}^k \quad k \in K,\ r = 1, \ldots, R,\ (i, j) \in A \setminus \{(O_r, O_{r+1})\}, \tag{7}$$

$$t_{ir}^k - M_{ij}^k \left(1 - x_{ijr}^k\right) \le t_{jr}^k \quad k \in K,\ r = 1, \ldots, R,\ (i, j) = (O_r, O_{r+1}), \tag{8}$$

$$\sum_{i \in N} \left[\frac{D_i}{Q_k} \left(\sum_{j \in N \cup \{O_{r+1}\}} x_{ijr}^k \right) \right] \le 1 \quad k \in K,\ r = 1, \ldots, R, \tag{9}$$

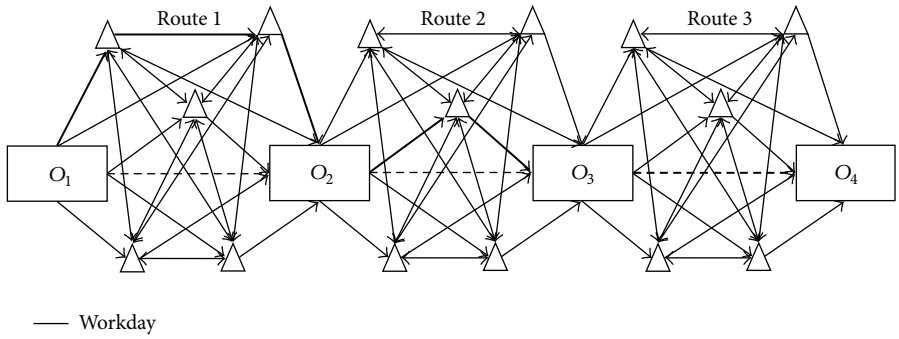

Figure 1: Example of a digraph G regarding one vehicle and its workday.

$$t^k_{O_{R+1}R} - t^k_{O_1,1} \leq W_{\max,k} \quad k \in K, \tag{10}$$

$$x^k_{ijr} \in \{0,1\} \quad k \in K,\ (i,j) \in A,\ r = 1,\ldots,R, \tag{11}$$

$$t^k_{ir} \geq 0 \quad k \in K,\ r = 1,\ldots,R,\ i \in N \cup \{O_1,\ldots,O_{R+1}\}. \tag{12}$$

Constraint (1) imposes that each customer i must be visited only once by a vehicle k in a route r. According to constraint (2), unauthorised vehicles are kept from visiting customers that have accessibility restrictions. In (3), it is imposed that every vehicle k must leave from each route origin. Analogously, constraint (4) imposes that every vehicle k must reach each route destination. Constraint (5) ensures the flow conservation for a customer. Constraint (6) imposes that time windows must be respected. The group of constraints (7) and (8) establishes the relationship between the vehicle departure time from a customer and that customer's immediate successor. Constraint (9) states that a vehicle can only be loaded up to its capacity. Constraint (10) guarantees that the workday duration of a vehicle does not exceed its limit $W_{\max,k}$. The group of constraints (11) and (12) defines the solution space of the decision variables.

3.1. Multiple Route Cuts. It is important to notice that the existence of the fictitious arc (O_r, O_{r+1}) when $R \geq 2$ is responsible for the existence of solutions that are equivalent from a physical point of view but distinct mathematically.

Equivalent solutions of a current solution exist when $R \geq 2$ and at least one workday of any vehicle travels at least one fictitious arc and there is, at least, one route serving any customer. In these cases, there is freedom in rearranging fictitious routes inside the workday without changing the sequence of routes serving customers, so as to generate equivalent workdays and, therefore, equivalent solutions.

In order to avoid the existence of equivalent solutions and reduce the expansion of the solution space when the value of R increases, valid inequalities (13) were derived:

$$x^k_{O_{r+1}O_{r+2}r+1} \geq x^k_{O_rO_{r+1}r} \quad k \in K,\ r = 1,\ldots,(R-1). \tag{13}$$

According to constraints (13), applicable only when $R \geq 2$, if a vehicle k travels the fictitious arc (O_r, O_{r+1}) along route r, that is, if $x^k_{O_rO_{r+1}r} = 1$, then the fictitious arcs of the following routes must be travelled.

4. Resolution Algorithm

To solve the problem described in Section 2 and formulated in Section 3, a heuristic procedure to find good valid integer solutions and a column generation procedure to find dual bounds are proposed. These procedures are described in the following subsections.

4.1. Constructive Heuristic. In summary, the developed procedure inserts customers into vehicle workdays and builds an initial solution after inserting all of them according to the pseudocode Constructive_Heuristic (See Algorithm 1).

4.2. Tabu Search. Tabu search is a metaheuristic with a wide range of applications in vehicle routing problems [17] that combines intensification and diversification strategies using short- and long-term memories. For the present problem, the removal of a customer from the workday of a given vehicle and the insertion of that customer in the workday of another vehicle, as well as a move to reposition a customer within a vehicle's workday, are both used as moves to find neighbour solutions. In addition, rather than employing long-term memory to help with the diversification process, a predefined number of random moves are performed, enough to generate a solution distant from the current solution. The aspiration criterion adopted is the acceptance of a tabu solution when it globally improves the objective function and when its cost is lower than the best nontabu solution.

The major parameters for calibrating the implemented tabu search are size of the short-term memory, number of iterations without minimum local solution improvement that triggers diversification, number of random moves to diversify a solution, and maximum number of diversifications.

In each iteration, all possible moves are verified instead of stopping the neighbourhood search when a solution whose cost is lower than the current solution cost is found. The latter approach is widely used in the literature when many moves are considered and the computational cost of searching in the entire neighbourhood of a solution is high. However, in this study only two simple moves are used, and the entire neighbourhood is verified. This approach has been proven effective, as described in Section 5.

The tabu activation rule is based on the two move attributes: customer and destination vehicle. If there is a move

```
Constructive_Heuristic( )
(1) find customers' geometric centre
(2) divide customers into 4 quadrants with their geometric centre as a reference
(3) sort each quadrant list of customers in increasing order of time window opening
(4) //Travel quadrants in anti-clockwise sequence starting from the upper right quadrant//
(5) for each quadrant customer list do
(6)         for every customer with a list of authorised vehicles do
(7)                 find the smallest insertion cost position among authorised vehicle workdays
(8)                 if a position was found then
(9)                         insert the customer and remove it from the quadrant list
(10) sort the list of vehicles in decreasing order of cargo capacity
(11) sort the list of vehicles in increasing order of cost with a stable sorting algorithm
(12) for each vehicle in the sorted list of vehicles do
(13)         //Travel quadrants in an anti-clockwise sequence//
(14)         for each quadrant customer list do
(15)                 for each customer do
(16)                         find smallest insertion cost position in the current vehicle workday
(17)                         if a position was found then
(18)                         insert the customer and remove it from quadrant list
(19) //Check if the current solution is infeasible//
(20) if there are unserved customers then
(21)         if tabu search was never called then
(22)                 //"Make room" for future insertions by cost reduction//
(23)                 call tabu search of few iterations to improve the current solution
(24)                 go to line 5 to insert unserved customers
(25)         else
(26)                 call a commercial software to find a feasible solution
```

Algorithm 1: Constructive_Heuristic Algorithm—procedure to find a feasible solution.

that has removed the customer from the destination vehicle workday in the short-term memory, then the reinsertion of the customer is tabu. Otherwise, the insertion is not tabu.

Furthermore, in the present implementation, when a move fails to improve the minimum local solution, it is verified whether the maximum number of iterations without minimum local solution improvement has been reached. In this case, a diversification will be executed. Diversification is carried out by randomly choosing customer reallocation moves between vehicles. These moves are executed until the specified number of random moves is reached. The tabu search ends when the maximum number of diversifications is reached.

4.3. Extensive Formulation and Column Generation. It is possible to provide an alternative formulation to the routing problem after observing that (IP) has primal block diagonal structure. This structure is composed by linking constraint (1) and independent constraint blocks from (2) to (13) for each vehicle k, corresponding to workdays. Let P^k be the set of all feasible workdays p of vehicle k. The alternative mathematical formulation of the problem, denoted as master problem (MP), is as follows:

$$\text{Min} \sum_{k \in K} \sum_{p \in P^k} c_p^k \cdot \lambda_p^k, \tag{MP}$$

s.t.

$$\sum_{k \in K} \sum_{p \in P^k} a_{ip}^k \cdot \lambda_p^k = 1 \quad i \in N, \tag{14}$$

$$\sum_{p \in P^k} \lambda_p^k = 1 \quad k \in K, \tag{15}$$

$$\lambda_p^k \in \{0, 1\} \quad k \in K,\ p \in P^k, \tag{16}$$

where λ_p^k is a binary decision variable equal to 1 if and only if workday p is chosen for vehicle k and 0 otherwise; $c_p^k = \sum_{r=1}^{R} \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijrp}^k$ for $k \in K$, $p \in P^k$ is the cost of workday p of vehicle k; $a_{ip}^k = \sum_{r=1}^{R} \sum_{j \in N \cup \{O_{r+1}\}} x_{ijrp}^k$ for $k \in K$, $i \in N$, $p \in P^k$ is the number of times customer i is visited by vehicle k on workday p; $x_{ijrp}^k \in \{0, 1\}$ for $k \in K$, $r = 1, \ldots, R$, $(i, j) \in A$, $p \in P^k$ is equal to 1 if vehicle k along route r travels the arc (i, j) on workday p; $x_{ijrp}^k = 0$ otherwise.

Constraints (14) impose that each customer must be served only once, constraints (15) impose that exactly one duty is selected for each vehicle, and constraints (16) ensure that the solution is integer.

We remark that (MP) has an exponential number of variables. Therefore, we resort to column generation [18] to solve the linear relaxation of the master problem (LMP). We further remark that LMP is equivalent to the Dantzig Wolfe reformulation operated on constraints (2)–(13) of (IP)

where the resulting constraints (14) are the so-called linking constraints and constraints (15) are the convexity constraints.

In LMP, the set partitioning constraints (14) are replaced with set covering constraints: $\sum_{k\in K}\sum_{p\in P^k} a_{ip}^k \cdot \lambda_p^k \geq 1$ for $i \in N$. When the triangular inequality holds on the cost matrix and load splitting is not allowed, no optimal solution visits any customer more than once. Consequently, the two formulations are equivalent. Set covering constraints are preferable to set partitioning constraints because the associated dual variables are restricted to assume nonnegative values.

To solve LMP, we apply column generation. Therefore, we consider a subset of workdays for each vehicle obtaining a restricted master problem (LRMP) so that a feasible solution to LMP exists. We solve the LMP and consider dual variables π and σ associated with constraints (14) and (15), respectively. We solve a pricing subproblem for every vehicle k and iterate the algorithm until no column with negative reduced cost exists for every vehicle.

The pricing subproblem consists of finding a workday that potentially contributes to reducing the objective function of LRMP; that is, a workday whose reduced cost for the new variable λ_p^k is negative. The pricing subproblem, denoted as SP, has the objective function $\text{Min}\sum_{r=1}^{R}\sum_{i\in N\cup\{O_r\}}\sum_{j\in N\cup\{O_{r+1}\}}(c_{ijr}^k-\pi_i)\cdot x_{ijr}^k-\sigma_k$, where $\pi_{O_r}=0$ for $r=1,\ldots,R$, subject to constraints (2) to (13) regarding vehicle k.

4.3.1. Initialisation. The LRMP is initialised with a feasible solution by calling the constructive heuristic, followed by a tabu search of few iterations to obtain a good initial solution at a low computational cost. In most cases, the number of columns generated for LRMP optimality is significantly reduced with a good initial solution (good initial set of columns).

4.3.2. Lower Bounding and Anticipated Finalisation. In some cases, it is possible to interrupt the column generation scheme before obtaining the optimal solution with the calculation of a lower bound, LB_{CG}, according to the expression $\text{LB}_{\text{CG}} = Z_{\text{LRMP}} + \sum_{k=1}^{|K|} Z_{\text{SP},k}$, where $Z_{\text{SP},k}$ is the optimal solution or a lower bound of the subproblem of vehicle k. A detailed deduction of the expression of the lower bound in the column generation technique can be obtained in the tutorial chapter of Desrosiers and Lübbecke [19].

4.3.3. Stabilisation. A well-known column generation property is that dual variable values do not smoothly converge to their respective optima but significantly oscillate with seemingly no regular pattern. This behaviour is undesirable, as it represents a major efficiency issue.

Several stabilisation methods that attempt to accelerate convergence have been proposed in the column generation literature.

Two different and widely used stabilisation techniques, namely, the interior point technique described by Rousseau et al. [20] and the method commonly referred to as BoxPen stabilisation, proposed by du Merle et al. [21], were implemented and compared with regard to the present problem.

The interior point technique is a method that addresses degeneracy and convergence difficulties by selecting a dual solution inside the optimal space rather than retrieving an extreme point. To achieve the centralisation of dual values, several extreme points of the optimal dual polyhedron are generated, and the interior point is computed as a convex combination of these extreme points.

The BoxPen method, proposed by du Merle et al. [21], combines two stabilisation techniques. The first technique, described by Marsten et al. [22], consists of defining a box around the previous dual solution and modifying the master problem so that the feasible dual space is limited to the area defined by these boxes. The second technique, proposed by Kim et al. [23], is to adapt the master problem so that the distance separating a dual solution from the previous dual solution is linearly penalised. In summary, the BoxPen method imposes soft limits on the dual variables and a penalty in the objective when dual variables take values outside of these limits (box).

Computational results have shown that the BoxPen method typically requires fewer iterations and a lower overall computing time to solve the LMP when compared to the interior point stabilisation technique, and therefore, the stabilisation method was chosen for the present problem.

4.4. Pricing Subproblem. The pricing subproblem SP can be cast as the computation of an elementary minimum cost path considering resource constraints (ESPPRC) in digraph G, in which the costs in the arcs are modified according to the shadow prices of the constraints (14) of the LRMP. A common and widely employed technique for solving this problem is dynamic programming, as successfully used by Desrosiers et al. [24] and Feillet et al. [15].

It should be stressed that the real routing problem considered here has some particular constraints, such as the possibility of a vehicle travelling multiple routes per day and the limitation on vehicle driver working hours that require a particular treatment, that, to the best of our knowledge, have not been addressed to date in the literature by exact methods.

A shortest path problem where waiting time is undesired along the path, namely, the shortest path problem with waiting costs (SPWC), was introduced by Desaulniers et al. [25]. In the SPWC, the elapsed time between the arrival and the departure time at a node is referred to as waiting time and there is linear cost penalty that is imposed for each unit of time spent waiting along the path. Deasaulniers and Villeneuve [26] propose two alternative formulations of the SPWC for which algorithms already exist, namely, the shortest path problem with linear node costs and a two-resource generalised shortest path problem with resource constraints.

Unlike the SPWC, in the present problem, there is only the cost of travelling arcs in the objective function, and idle time must be minimised only enough to ensure path

feasibility with regard to the maximum workday duration along the minimum cost path.

4.4.1. Dynamic Programming Algorithm. The algorithm developed herein was based on the bounded bidirectional dynamic programming algorithm proposed by Righini and Salani [27, 28] that, in turn, is based on the algorithm developed by Desrochers and Soumis [29] to solve the resource constrained shortest path problem (RCSPP).

In the present problem, a particular resource called the workload resource was introduced. The workload resource is limited to $W_{\max,k}$. This resource consists of the time elapsed since the service start time of O_1 and its consumption depends on the arcs travelled, the sequence of vertices visited, and the service start time of O_1.

Idle time is generated whenever a vehicle arrives at j before the opening of its time window. In this case, the vehicle is forced to wait until a_j to start the service, and this waiting time is incorporated into the workload resource. In traditional dynamic programming algorithms for solving the ESPPRC [27], the O_1 service start time is always fixed at 0 with no implication for the final result, even though the mathematical formulation allows this variable to assume any positive value. In the present problem, the workload resource depends on the value of this variable. In short, whenever idle time is created, the dynamic programming algorithm has to recursively recalculate the service start times along the path to minimise the workload resource consumption.

However, the simple existence of the workload resource in a label is not sufficient for finding the optimal solution because there are situations in which a dominated label may be part of the optimal path. This behaviour occurs because of the capacity to serve the origin later and thus eliminate idle time along the route is smaller in the first case. Because of this feature, it is necessary to create an additional resource, the value of which represents the potential to eliminate idle time such that the idle time generated during the extension of the labels is minimised or even fully eliminated.

In Figure 2, it is possible to visualise an example with four customers in which $W_{\max,k} = 5$, $s^k_{DC} = 0$, and $[a_{DC}, b_{DC}] = [0, 10]$. Values t and c represent the time and cost of travelling an arc, respectively. Note that path DC-1-2-3 is apparently better than path DC-2-1-3 for reaching 3 at the same instant in time, with the same number of hours worked and with a lower cost by 1 unit. However, path DC-1-2-3 serves 4 with 5 units of workload consumed and is incapable of travelling arc 4-DC with cost $c = -10$. In contrast, path DC-2-1-3 is capable of leaving the origin at $t = 1$ and travelling path 3-4-DC, defining the optimal solution: DC-2-1-3-4-DC. The new resource proposed is initially (at the origin) equal to the planning horizon, and along the path, it is consumed progressively until it is exhausted or not at the final destination.

The vector of resources $\mathbf{R}$ is composed of the following components: q (consumed fraction of vehicle capacity), t (service start time in a forward label or departure time in a backward label), w (time consumed out of the working hours), Δ (idle time reducing capacity), S (visit vector), and vstQty (number of vertices served along the path). The label also has component r, which indicates the route to which the label belongs, and the cost C.

When a label $(\mathbf{R}, C, i)$ associated with vertex i is extended to vertex j generating the label $(\mathbf{R}', C', j)$, the consumption of each resource is updated according to the rules presented in the next sections, except for q, t, S and C, whose rules are the same as those described by Righini and Salani [28] and are thus not presented. It is worth highlighting that the extension along the fictitious route arc (O_r, O_{r+1}) is forbidden in both the forward and backward directions, and that the extension to j is only feasible if the vehicle is authorised to serve customer j.

4.4.2. Workload Resource. The calculation of w', the workload resource, for forward labels is initially given by $w' = \max\{w + (s^k_i + \tilde{t}^k_{ij}), w + (a_j - t)\}$. For backward labels, the value of w' is $w' = \max\{w + s^k_i + \tilde{t}^k_{ji}, w + \{[b_{DC} - (b_j + s_j)] - t\}\}$.

However, idle time is generated when the vehicle reaches vertex j before the opening of its time window, $t + s^k_i + \tilde{t}^k_{ij} < a_j$, in a forward extension or, when the vehicle departs from vertex j after the end of its departure time window, $t + s^k_i + \tilde{t}^k_{ji} \le b_{DC} - (b_j + s^k_j)$, in a backward extension. Based on the expressions above, the idle time is incorporated into the workday duration. To ensure future viability in relation to the hours worked on a path after adding new arcs, it is necessary to eliminate the generated idle time as much as possible. To do so, idle time, t_{idle}, must first be calculated either when reaching j before a_j in a forward extension, $t_{idle} = a_j - (t + s^k_i + \tilde{t}^k_{ij})$, or when leaving from j after $(b_j + s^k_j)$ in a backward extension, $t_{idle} = [b_{DC} - (b_j + s^k_j)] - (t + s^k_i + \tilde{t}^k_{ji})$.

Eliminating idle time in the forward direction is only possible when there is a margin to serve the origin later, displacing the service start time of all vertices previous to j along the path. For the backward direction, the elimination of idle time is only possible when there is a margin to "leave earlier from the final destination," displacing the departure time of all vertices previous to j along the path. This is verified in the forward direction by calculating the maximum displacement capacity of the origin service start time without making the path infeasible according to expression $\Delta' = \min\{\Delta_v \mid \Delta_v = (b_v - t_v),\ v \in \text{path}(\mathbf{R}, C, i)\}$. For the backward direction, $\Delta' = \min\{\Delta_v \mid \Delta_v = [b_{DC} - (a_v + s_v)] - t_v,\ v \in \text{path}(\mathbf{R}, C, i)\}$.

For the forward direction, after calculating t_{idle} and Δ', when $\Delta' > 0$ and $\Delta' \ge t_{idle}$, then $w' = w + s^k_i + \tilde{t}^k_{ij}$ and $t_v = t_v + t_{idle}$, for all $v \in \text{path}(\mathbf{R}, C, i)$; that is, t_{idle} units of Δ' are consumed. When $\Delta' > 0$ and $\Delta' < t_{idle}$, then $w' = w + s^k_i + \tilde{t}^k_{ij} + (t_{idle} - \Delta')$ and $t_v = t_v + \Delta'$, for all $v \in \text{path}(\mathbf{R}, C, i)$, that is all Δ' is consumed.

For the backward direction, after calculating t_{idle} and Δ', in the case $\Delta' > 0$ and $\Delta' \ge t_{idle}$, then $w' = w + s^k_i + \tilde{t}^k_{ji}$ and $t_v = t_v + t_{idle}$, for all $v \in \text{path}(\mathbf{R}, C, i)$. In the other case, when

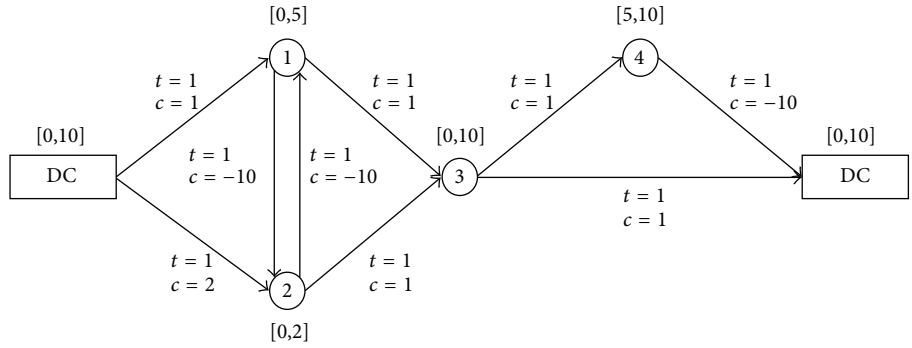

Figure 2: Example where the idle time reducing capacity resource is required.

$\Delta' > 0$ and $\Delta' < t_{\text{idle}}$, then $w' = w + s_i^k + \widetilde{t}_{ji}^k + (t_{\text{idle}} - \Delta')$ and $t_v = t_v + \Delta'$, for all $v \in \text{path}(\mathbf{R}, C, i)$.

The created forward or backward label is only feasible if $w' \leq W_{\max,k}$.

4.4.3. Idle-Time-Reducing Capacity Resource. The workload resource is not sufficient to fully identify a label in relation to the working hours and requires the use of a new resource that is capable of identifying the capacity of the path to reduce idle time, Δ. This resource is crucial for calculating working hours during the extension of a label so that the path feasibility is not lost after the incorporation of new arcs.

As Δ is progressively consumed along the path, Δ can be considered a new resource and updated according to the procedure described in Section 4.4.2 when idle time is generated during the extension. If idle time is not generated and the service start time or departure time does not violate the time window of j, then $\Delta' = \min\{\Delta, (b_j - t')\}$ for forward labels and $\Delta' = \min\{\Delta, [b_{\text{DC}} - (a_j + s_j^k)] - t'\}$ for backward labels.

Because $0 \leq \Delta' \leq b_{\text{DC}}$, there is no need to verify whether the extension feasibility is lost in relation to this resource.

4.4.4. Multiple Routes. If only one route is considered, the reduced digraph $G^* = (N \cup O_1 \cup O_2, A^*)$ is sufficient for the formulation. When more than one route is allowed up to a limit of R routes, the natural strategy is to replicate digraph G^*R times, obtaining digraph G in which a path from O_1 to O_{R+1} represents a workday. However, for computational implementation, it is not necessary to replicate digraph G^*R times because arcs linking destination O_2 to the origin O_1 and vice versa (backward direction) are used, together with specific extension rules for these arcs. For route control, a route r indicator is added to each label. At first, an initial forward label is added to origin O_1 with $r = 1$ and an initial backward label to destination O_2 with $r = R$.

Forward and backward extension procedures are executed until all of the forward and backward labels have been extended. At this moment, all of the forward and dominant labels in O_2 are extended to O_1 according to a special rule. The same occurs for the backward and dominant labels of O_1, which are extended to O_2 according to a special second rule. Among other things, during this extension, the route indicator r of the forward labels is incremented as the route indicator of the backward labels is decremented; thus, a new cycle for the application of the extension procedures begins.

This process is repeated until $r = R$ in the forward labels and $r = 1$ in the backward labels. The cost and time for travelling the arc to return to the beginning of a new route are equal to 0. The extension of a forward label from O_2 to O_1 is made according to the rules $C' = C$, $q' = 0$, $t' = t$, $w' = w$, $r' = r + 1$, $\Delta' = \Delta$, and

$$S'[v] = \begin{cases} 0, & v = O_2, \\ S[v], & v \neq O_2. \end{cases} \tag{17}$$

The extension of a backward label from O_1 to O_2 is made according to the rules $C' = C$, $q' = 0$, $r' = r - 1$, $\Delta' = \Delta$, and

$$S'[v] = \begin{cases} 0, & v = O_1, \\ S[v], & v \neq O_1. \end{cases} \tag{18}$$

If the backward label does not visit any customer along route r, then $t' = t$ and $w' = w$. If the backward label visits at least one customer along route r, then $t' = t + s_{\text{DC}}^k$ and $w' = w + s_{\text{DC}}^k$.

4.4.5. Lexicographic Ordering of Labels. The dominant labels are kept organised in increasing lexicographic order along all of the iterations as described in Ceselli et al. [14]. A label L_1 is considered lexicographically smaller than a label L_2 if $w_1 < w_2$ or $w_1 = w_2$ and $\text{vstQty}_1 < \text{vstQty}_2$. As the workload resource is minimised in a label, it is a significant resource for the dominance rule, and for this reason, it is selected as the major ordering criterion, followed by the number of vertices served. In short, whenever a vertex is selected so that its labels not yet treated are extended, the selection and extension of labels occur in increasing lexicographic order.

4.4.6. Bounding on Resource. Based on the strategy adopted by Righini and Salani [28] to prevent the proliferation of labels in both directions, a critical resource is selected such that labels consuming over 50% of the resource are eliminated from each of the two directions. In the present problem, the critical resource is time; thus, a label is discarded when $t' > 0.5 \cdot (b_{\text{DC}} - a_{\text{DC}})$ during an extension.

Table 1: Results obtained for R1 instances with the first 50 customers.

Instance	Customer qty.	Route qty.	Heuristic cost	Heuristic time (s)	Lower bound	Lower bound time (s)	Heuristic error (%)	Iterations
R101	50	1	967.497	235	955.531	2	1.237	45
R102	50	1	805.559	323	770.943	4	4.297	47
R103	50	1	657.448	375	642.114	20	2.332	84
R104	50	1	606.484	436	599.376	34	1.172	63
R105	50	1	742.452	283	722.445	4	2.695	86
R106	50	1	679.535	357	665.543	11	2.059	77
R107	50	1	604.177	394	598.596	17	0.924	63
R108	50	1	594.898	440	586.373	58	1.433	69
R109	50	1	624.043	342	623.274	5	0.123	65
R110	50	1	612.588	401	599.062	12	2.208	81
R111	50	1	606.677	402	601.543	13	0.846	56
R112	50	1	604.155	453	588.483	22	2.594	63
			675.459	**370**	**662.774**	**17**	**1.827**	**67**
R101	50	2	978.249	437	955.531	1	2.322	46
R102	50	2	796.028	549	770.479	6	3.210	85
R103	50	2	656.540	610	642.114	42	2.197	89
R104	50	2	607.629	683	599.214	197	1.385	71
R105	50	2	753.951	507	721.848	3	4.258	89
R106	50	2	677.576	598	663.198	20	2.122	103
R107	50	2	604.177	644	598.596	46	0.924	98
R108	50	2	596.087	647	584.634	255	1.921	89
R109	50	2	627.399	582	623.274	8	0.657	82
R110	50	2	612.588	652	599.062	23	2.208	91
R111	50	2	608.576	658	600.540	29	1.320	96
R112	50	2	599.212	710	588.442	46	1.797	66
			676.501	**606**	**662.244**	**56**	**2.027**	**84**
R101	50	3	962.944	549	955.531	1	0.770	41
R102	50	3	803.934	679	770.479	5	4.161	49
R103	50	3	656.540	761	642.114	60	2.197	92
R104	50	3	608.651	844	599.214	314	1.550	91
R105	50	3	742.080	646	721.848	4	2.726	93
R106	50	3	669.851	750	663.198	25	0.993	100
R107	50	3	604.177	811	598.596	53	0.924	67
R108	50	3	598.581	796	584.634	432	2.330	94
R109	50	3	625.534	733	623.274	10	0.361	50
R110	50	3	613.450	816	599.062	33	2.345	89
R111	50	3	606.677	823	600.540	42	1.012	92
R112	50	3	601.904	893	588.442	75	2.237	67
			674.527	**758**	**662.244**	**88**	**1.801**	**77**

4.4.7. Dominance Rule. The efficiency of the dynamic programming algorithm depends mainly on its capacity to identify and to eliminate labels that will certainly not be a part of the optimal solution. This prevents these undesired labels from generating an unnecessary proliferation of labels that largely contribute to increasing the total computational cost. To conduct this identification and elimination of labels that are certainly not a part of the optimal solution, there are dominance tests that are always executed when a label is extended so that the labels at each vertex are nondominated.

According to the dominance rule, label L_1 dominates label L_2 if all of the following expressions are true: $C_1 \leq C_2$;

Table 2: Results obtained for R2 instances with the first 50 customers.

Instance	Customer qty.	Route qty.	Heuristic cost	Heuristic time (s)	Lower bound	Lower bound time (s)	Heuristic error (%)	Iterations
R201	50	1	629.643	323	617.533	3	1.923	56
R202	50	1	580.792	388	540.406	758	6.954	60
R203	50	1	542.673	892	515.961	33367	4.922	88
R204	50	1	504.971	1050	*	*	*	*
R205	50	1	582.809	423	563.371	41	3.335	83
R206	50	1	548.465	541	529.240	520	3.505	83
R207	50	1	522.382	557	*	*	*	*
R208	50	1	508.412	1090	*	*	*	*
R209	50	1	543.668	833	526.939	814	3.077	75
R210	50	1	556.702	532	532.627	493	4.325	76
R211	50	1	514.903	985	489.296	4843	4.973	92
			548.675	**692**	**539.422**	**5105**	**4.127**	**77**
R201	50	2	631.263	521	617.533	6	2.175	83
R202	50	2	584.234	603	552.827	2712	5.376	88
R203	50	2	547.387	726	*	*	*	*
R204	50	2	516.083	832	*	*	*	*
R205	50	2	585.736	660	563.292	213	3.832	90
R206	50	2	546.744	680	527.546	5298	3.511	92
R207	50	2	523.277	771	*	*	*	*
R208	50	2	509.368	851	*	*	*	*
R209	50	2	552.351	1224	519.866	2944	5.881	94
R210	50	2	559.125	655	530.574	2736	5.106	87
R211	50	2	524.933	776	491.795	22175	6.313	105
			552.773	**754**	**543.348**	**5155**	**4.599**	**91**
R201	50	3	637.648	655	617.533	18	3.155	91
R202	50	3	581.999	756	555.689	8441	4.521	91
R203	50	3	546.259	871	*	*	*	*
R204	50	3	514.158	1002	*	*	*	*
R205	50	3	585.951	823	562.820	1051	3.948	84
R206	50	3	552.396	825	526.078	25137	4.764	95
R207	50	3	531.691	880	*	*	*	*
R208	50	3	507.904	1039	*	*	*	*
R209	50	3	550.108	854	517.297	7351	5.964	101
R210	50	3	559.496	859	534.958	18095	4.386	91
R211	50	3	524.667	954	494.806	92018	5.691	109
			553.843	**865**	**544.169**	**21730**	**4.633**	**95**

Value of cells marked with * were not computed due to an overall time greater than 93000 s.

$q_1 \leq q_2$; $t_1 \leq t_2$; $w_1 \leq w_2$; $t_1 - w_1 \geq t_2 - w_2$; $\Delta_1 \geq \Delta_2$; $r_1 = r_2$; $\text{vstQty}_1 \leq \text{vstQty}_2$ and $S_1[v] \leq S_2[v]$, for all $v \in \{N \cup \{O_1, O_2\}\}$.

4.4.8. Heuristic Labelling Algorithm. Aiming at increasing the convergence speed of the column generation technique, columns are added by a heuristic labelling algorithm until no new column with negative reduced cost is found. The subproblem is then solved to optimality by the exact labelling algorithm.

The exact labelling algorithm is typically made heuristic by removing from its dominance rule combinatorial nature resources, which reduces the number of labels generated, as described by Ceselli et al. [14]. In the present study, the Heuristic Labelling Algorithm uses a relaxed dominance rule that does not have the visit vector, S, the capacity to reduce idle time, Δ, and component $(t - w)$.

4.4.9. Forward-Backward Joint Feasibility Tests and Search for Optimal Solution. After the end of the label extension process

(1) $r_1 \leq r_2$;
(2) There is arc (i, j);
(3) $S_1[v] + S_2[v] \leq 1 \ \forall v \in N$
(4) $q_1 + q_2 \leq 1$, that is, total capacity is less than or equal to 100% of Q_k;
(5) $t_i + s_i + \tilde{t}_{ij}^k + t_j \leq b_{DC}$;
(6) If $R > 1$, $i = O_1$ and $j = O_2$ then
    $r_2 = R$ must be true.
    End
(7) After calculating $t_{idle} = \left[b_{DC} - \left(t_j + s_j\right)\right] - \left(t_i + s_i + \tilde{t}_{ij}^k\right)$
    $t_{idle} \geq 0$ must be true.
    If $(t_{idle} = 0)$, then $w_i + w_j + \left[\left(b_{DC} - t_j\right) - t_i\right] \leq W_{max,k}$ must be true.
    Else
        Let $\Delta = \Delta_1 + \Delta_2$
        If $(\Delta \geq t_{idle})$, then $w_i + w_j + \left[\left(b_{DC} - t_j\right) - t_i\right] - t_{idle} \leq W_{max,k}$ must be true.
        Else, $w_i + w_j + \left[\left(b_{DC} - t_j\right) - t_i\right] - \Delta \leq W_{max,k}$ must be true.
        End
    End

Algorithm 2: Feasibility criteria for combining forward and backward labels.

for all of the routes, the forward labels are combined with the backward labels to form paths from O_1 to O_{R+1}. The optimal solution is determined after all of the feasible combinations between a forward label and backward label at each possible pair of vertices (i, j) are verified. For this purpose, there are seven tests that are applied to ensure that the combination of the two labels, L_1 forward in i and L_2 backward in j, creates a feasible path from O_1 to O_{R+1}. The seven tests are described in Algorithm 2.

Cost C of the combination between labels L_1 and L_2 is calculated as $C = C_1 + C_2 + c_{ij}$.

When searching for an optimal solution, a very simple but rather efficient approach to reduce the computational cost is to first compute the cost of the label combination and then test the joint feasibility, only if its cost is lower than the lowest cost already found.

4.4.10. State-Space Relaxation and Decremental State-Space Relaxation. State-space relaxation was introduced by Christofides et al. [30]. The state-space, Ψ, explored by the exact dynamic programming algorithm is projected onto a lower dimensional space, Γ, so that each state in Γ retains the minimum cost among those of its corresponding states in Ψ (assuming the objective function must be minimised). In space Γ, the number of states to be explored is drastically reduced; the drawback is that some original state corresponding to an infeasible solution in Ψ may be projected onto a state corresponding to a feasible solution in Γ; therefore, the search in the relaxed state-space does not guarantee finding an optimal solution but rather a lower bound.

The state-space relaxation algorithm considered in this study consists of mapping each state $(\mathbf{R}, C, r, i)$ onto a new state $(\widehat{\mathbf{R}}, C, r, i)$ in which the visit vector S is removed from the resource vector $\mathbf{R}$ by being mapped as the existing resource vstQty (vstQty $= \sum_{i \in N \cup \{O_1, O_2\}} S[i]$). Except for the removal of the visit vector S, there are no differences when compared to the exact dynamic programming algorithm dominance rule.

The considered mapping makes a large difference with respect to the exact dynamic programming algorithm in which the visit vector S yields an exponential number of possible states. Because the state (label) does no longer keeps information about the set of already visited vertices, cycles are allowed; therefore, the path is guaranteed to be feasible with respect to the resource constraints but it is not guaranteed to be elementary.

As opposed to the state-space relaxation, the decremental state-space relaxation pursues a compromise between forbidding and allowing multiple visits to the vertices, that is, a compromise between the exact dynamic programming algorithm and state-space relaxation algorithm. This compromise is accomplished by identifying some vertices as critical according to the solution obtained and by preventing multiple visits to critical vertices in subsequent runs while allowing multiple visits to noncritical vertices.

In summary, the decremental state-space relaxation algorithm runs iteratively the state-space relaxation procedure in which each label also has a binary vector $\widehat{S}$ playing the same role as S in the exact dynamic programming algorithm: if $\widehat{S}[j] = 1$, then j is critical, and $\widehat{S}[j] = 0$ otherwise. Consequently, the extension of a label $(\widehat{\mathbf{R}}, C, r, i)$ from vertex i to vertex j is feasible with regard to $\widehat{S}$ only if $\widehat{S}[j] = 0$ or if $\widehat{S}[j] = 1$ and j is not visited along the path $(\widehat{\mathbf{R}}, C, r, i)$.

Every time a solution with cycles is produced by the modified state-space relaxation procedure, the vertices visited more than once are marked as critical, and the algorithm restarts. The optimal solution, which is guaranteed to be feasible with respect to the resource constraints and to be elementary, is obtained when a solution with no cycles is produced.

It is worth highlighting that the decremental state-space relaxation is used only when optimality is required, that is,

only when the dominance rule is complete (includes the visit vector S). The heuristic labelling algorithm is based on the regular bidirectional dynamic programming algorithm.

5. Computational Results

Solomon's benchmark vehicle routing problem instances R1 and R2 were adapted to the context of the problem studied.

The maximum workday duration for any driver is $\lfloor (9.5/16) \cdot (b_{DC} - a_{DC}) \rfloor$, where $[a_{DC}, b_{DC}]$ is the depot time window, 9.5 hours is the maximum workday duration in the real context analysed and 16 hours is the real delivery operation planning horizon.

Based on Solomon's instance service time at each customer, defined as s_i, and considering a depot loading time $s_{DC} = 10$, then the service/loading times are computed as $s_i^k = 1.5s_i$ for a truck, $s_i^k = 1.25s_i$ for a LCV, and $s_i^k = s_i$ for a van.

In every instance, customer 2 can only be served by a specific truck, and customer 3 can only be served by another specific truck.

The tabu search executed is defined by the parameters as follows:

(1) short-term memory size = 20;

(2) number of iterations without improvement that triggers diversification = 100;

(3) number of diversifications = 1000;

(4) number of random moves to diversify a solution = 60.

Regarding the implemented BoxPen stabilisation method, the linear penalty for leaving the box is initially set to 0.001, and the box size around each dual variable is initially set to 0.2. If a subproblem is able to find a negative reduced cost path, then the linear penalty is increased by 10%. However, when there are no more such columns, the penalty is significantly decreased (divided by 1000), and every dual box is re-centred on the corresponding last dual value. This procedure guarantees column generation convergence. Furthermore, the initial dual solution is defined with potentially good initial dual values based on some knowledge of the problem, and consequently, they are estimated and computed as $\pi_i = \min_{j \in N}\{d_{ij}\}$ and $\sigma_k = VC_k \cdot \min_{j \in N}\{d_{O_k j}\}$.

In the available fleet for R1 instances there are 25 vehicles: 15 trucks, 5 LCVs, and 5 vans. In the available fleet for R2 instances there are 15 vehicles: 5 trucks, 5 LCVs, and 5 vans. The capacities of a truck, LCV, and van are 200 (as in Solomon's instances), 120 and 40, respectively. The costs per unit of distance run by a truck, LCV, and van are 1, 0.75 and 0.5, respectively. The average speeds of a truck, LCV and van are 1, 1.333 and 1.666, respectively.

The adapted R1 and R2 instances with the first 50 customers were solved with 1, 2, and 3 routes in a workday. The final results obtained are presented in Tables 1 and 2 with the corresponding lower bound cells filled with the LRMP optimal solution for R1 instances and with the first valid lower bound, that is, after one iteration with guaranteed optimality on subproblems, for R2 instances.

In every instance, the Cplex was not required to find a feasible solution.

A workstation with an Intel Core i7 2.80 GHz processor and 16374-Mb RAM memory was used to run the program developed in Java. CPLEX 12.2 is used to solve each LRMP and find a feasible initial solution when the constructive heuristic fails to find one.

6. Conclusions

In this paper, we presented a mathematical formulation based on arc flow variables that is able to solve a complex vehicle routing and scheduling problem. Valid inequalities were also presented to strengthen the formulation when more than one route is allowed in a workday.

To find integer solutions, a constructive heuristic and a metaheuristic based on tabu search were developed that can generate good solutions with an average error of less than 2.1% for R1 problems and 4.7% for R2 problems. It should also be noted that the heuristic computational cost is low and that the standard deviation of heuristic resolution times for instances with one, two, or three routes in a workday is also very low, which makes the heuristic solution procedure robust to a daily routing operation.

A slight increase in the average heuristic solution cost was observed when the number of routes is incremented due to an enlargement of the solution space without an increase in the number of diversifications and/or use of more sophisticated tabu search intensification and diversification strategies.

To evaluate the performance of the heuristic, a column generation algorithm with state-of-the-art techniques was presented that was able to generate excellent dual bounds. To solve the column generation pricing subproblem, a particular dynamic programming algorithm for the elementary shortest path problem with resource constraints that can address multiple routes in a workday with limitation on driver work hours was presented.

Future work related to this study may include the development of a branch-and-price algorithm, based on the column generation procedure proposed, to solve the problem exactly. Valid inequalities may also be derived to develop a branch-and-cut-and-price algorithm. Moreover, a branch-and-bound algorithm to solve to optimality the elementary shortest path problem with resource constraints may also be developed, by exploiting the lower bound given by the bounded bidirectional dynamic programming algorithm with state-space relaxation.

Acknowledgments

Financial support for this work was provided by CAPES and CNPq, Grant no. 558025/2009-9. This support is gratefully acknowledged. The authors also thank the anonymous referees for their useful references and constructive feedback.

References

[1] R. H. Ballou, *Business Logistics/Supply Chain Management*, Prentice-Hall, Englewood Cliffs, NJ, USA, 5th edition, 2004.

[2] M. M. Solomon and J. Desrosiers, "Time window constrained routing and scheduling problems," *Transportation Science*, vol. 22, no. 1, pp. 1–13, 1988.

[3] M. Desrochers, J. K. Lenstra, M. W. P. Savelsbergh, and F. Soumis, "Vehicle routing with time windows: optimization and approximation," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds., vol. 16 of *Studies in Management Science and Systems*, pp. 65–84, North-Holland, Amsterdam, The Netherlands, 1988.

[4] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992.

[5] É. D. Taillard, G. Laporte, and M. Gendreau, "Vehicle routeing with multiple use of vehicles," *Journal of the Operational Research Society*, vol. 47, no. 8, pp. 1065–1070, 1996.

[6] J. C. S. Brandão and A. Mercer, "The multi-trip vehicle routing problem," *Journal of the Operational Research Society*, vol. 49, pp. 799–805, 1998.

[7] A. Olivera and O. Viera, "Adaptive memory programming for the vehicle routing problem with multiple trips," *Computers & Operations Research*, vol. 34, no. 1, pp. 28–47, 2007.

[8] R. J. Petch and S. Salhi, "A multi-phase constructive heuristic for the vehicle routing problem with multiple trips," *Discrete Applied Mathematics*, vol. 133, no. 1–3, pp. 69–92, 2003.

[9] S. Salhi and R. J. Petch, "A GA based heuristic for the vehicle routing problem with multiple trips," *Journal of Mathematical Modelling and Algorithms*, vol. 6, no. 4, pp. 591–613, 2007.

[10] N. Azi, M. Gendreau, and J.-Y. Potvin, "An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles," *European Journal of Operational Research*, vol. 202, no. 3, pp. 756–763, 2010.

[11] N. Azi, M. Gendreau, and J.-Y. Potvin, "An exact algorithm for a single-vehicle routing problem with time windows and multiple routes," *European Journal of Operational Research*, vol. 178, no. 3, pp. 755–766, 2007.

[12] J. F. Cordeau, G. Laporte, and A. Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows," *Journal of the Operational Research Society*, vol. 52, no. 8, pp. 928–936, 2001.

[13] J. F. Cordeau and G. Laporte, "A tabu search heuristic for the static multi-vehicle dial-a-ride problem," *Transportation Research B: Methodological*, vol. 37, no. 6, pp. 579–594, 2003.

[14] A. Ceselli, G. Righini, and M. Salani, "A column generation algorithm for a rich vehicle-routing problem," *Transportation Science*, vol. 43, no. 1, pp. 56–69, 2009.

[15] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, "An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems," *Networks*, vol. 44, no. 3, pp. 216–229, 2004.

[16] J. K. Lenstra and A. H. G. Rinnooy Kan, "Complexity of the vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[17] O. Bräsy and M. Gendreau, "Vehicle routing with time windows, part II: metaheuristics," *Transportation Science*, vol. 39, pp. 119–139, 2005.

[18] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[19] J. Desrosiers and M. E. Lübbecke, "A primer in column generation," in *Column Generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds., GERAD 25th Anniversary Series, chapter 1, Springer, New York, NY, USA, 2005.

[20] L.-M. Rousseau, M. Gendreau, and D. Feillet, "Interior point stabilization for column generation," *Operations Research Letters*, vol. 35, no. 5, pp. 660–668, 2007.

[21] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen, "Stabilized column generation," *Discrete Mathematics*, vol. 194, no. 1–3, pp. 229–237, 1999.

[22] R. E. Marsten, W. W. Hogan, and J. W. Blankenship, "The BOXSTEP method for large-scale optimization," *Operations Research*, vol. 23, no. 3, pp. 389–405, 1975.

[23] S. Kim, K.-N. Chang, and J.-Y. Lee, "A descent method with linear programming subproblems for nondifferentiable convex optimization," *Mathematical Programming*, vol. 71, no. 1, pp. 17–28, 1995.

[24] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis, "Time constrained routing and scheduling," in *Network Routing*, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, Eds., vol. 8 of *Handbooks in Operations Research and Management Science*, pp. 35–139, North-Holland, Amsterdam, The Netherlands, 1995.

[25] G. Desaulniers, J. Lavigne, and F. Soumis, "Multi-depot vehicle scheduling problems with time windows and waiting costs," *European Journal of Operational Research*, vol. 111, no. 3, pp. 479–494, 1998.

[26] G. Desaulniers and D. Villeneuve, "Shortest path problem with time windows and linear waiting costs," *Transportation Science*, vol. 34, no. 3, pp. 312–319, 2000.

[27] G. Righini and M. Salani, "Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints," *Discrete Optimization*, vol. 3, no. 3, pp. 255–273, 2006.

[28] G. Righini and M. Salani, "New dynamic programming algorithms for the resource constrained elementary shortest path problem," *Networks*, vol. 51, no. 3, pp. 155–170, 2008.

[29] M. Desrochers and F. Soumis, "A generalized permanent labelling algorithm for the shortest path problem with time windows," *Information Systems Research*, vol. 26, pp. 193–214, 1988.

[30] N. Christofides, A. Mingozzi, and P. Toth, "State-space relaxation procedures for the computation of bounds to routing problems," *Networks*, vol. 11, no. 2, pp. 145–164, 1981.