

A branch-and-price algorithm for the vehicle routing problem with time windows on a road network

Citation for published version (APA):

Ben Ticha, H., Absi, N., Feillet, D., Quilliot, A., & van Woensel, T. (2019). A branch-and-price algorithm for the vehicle routing problem with time windows on a road network. *Networks*, 73(4), 401-417.
<https://doi.org/10.1002/net.21852>

Document license:
TAVERNE

DOI:
[10.1002/net.21852](https://doi.org/10.1002/net.21852)

Document status and date:
Published: 07/05/2019

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

A branch-and-price algorithm for the vehicle routing problem with time windows on a road network

Hamza Ben Ticha¹  | Nabil Absi¹ | Dominique Feillet¹  | Alain Quilliot² | Tom Van Woensel³

¹Ecole des Mines de Saint-Etienne and UMR CNRS 6158 LIMOS, CMP Georges Charpak, Gardanne, France

²LIMOS, UMR CNRS 6158 and ISIMA, Campus des Cézeaux, Aubière Cedex, France

³School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands

Correspondence

Hamza B. Ticha, Ecole des Mines de Saint-Etienne and UMR CNRS 6158 LIMOS, CMP Georges Charpak F-13541 Gardanne, France.
Email: hamza.ben-ticha@emse.fr

Funding information

This research was supported by the Conseil Régional d'Auvergne, European Fund for Regional Development (FEDER Auvergne region), Labex IMobS3.

Vehicle routing problems (VRP) concern the pickup and/or the delivery of goods from/to customers with vehicles. In the literature, most approaches consider the road network implicitly. Specifically, so-called customer-based graphs are used where nodes represent customers (plus the depot) and arcs represent best paths between customers. This model can affect solution quality when several attributes are defined on road segments (like travel time and distance). To handle that, two approaches are proposed in the literature. The road network can be represented using a multigraph that extends the customer-based graph and where an arc is introduced for every efficient path between two nodes. Alternatively, the problem can be solved on a graph that mimics the original road network. In this paper, we investigate the latter approach. We consider the VRP with time windows (VRPTW) and we develop a branch-and-price scheme. An extensive computational study based on several types of instances is conducted in order to evaluate this approach compared to the multigraph-based approach. As far as we know, our branch-and-price scheme is the first exact method for the VRPTW with the road-network model. Also, our computational study provides the first comparison between the two models: multigraph and road-network.

KEYWORDS

branch-and-price, multigraph, road-network graph, vehicle routing problems

1 | INTRODUCTION

Vehicle routing problems (VRP) aim to design an optimal set of routes to be used by a fleet of vehicles to visit a set of geographically dispersed customers. These routes must start and end at a depot and satisfy a set of constraints (vehicle capacity, customer's time windows, route duration, etc.). Since the introduction of the first VRP by Dantzig and Ramser [7], hundreds of papers and books have been devoted to these problems [14]. Many variants have been proposed to address the numerous issues that arise in real-life applications, such as the VRP with time windows (VRPTW) where transportation plans are constrained to satisfy customer requests within their time windows [19], the multidepot VRP where vehicles are based at different depots [6], and so on. Extensive reviews on the most common variants of the VRP are available in [12, 21].

Conventionally, VRP are tackled using a simple graph, abstracting the underlying road network. In this graph, called *customer-based graph*, a node is introduced for each point of interest (customer, depot, etc.) and an arc represents the best path in the road network between two nodes. This model implicitly assumes that these best paths can be defined a priori. Yet, in practice several attributes can be given on road segments (e.g., travel time, travel cost, energy consumption, etc.). Thus, alternative paths exist between pairs of points of interest. Not considering these alternatives may discard potentially good solutions from the solution space and may have a negative effect on the solution quality.

In the literature, an increasing number of papers investigate the effects of using the customer-based graph on the solution quality for VRP when several attributes are defined on road segments [4]. Two modeling approaches have been proposed to handle these effects. The first approach consists in representing the road network with a *multigraph* that extends the customer-based graph by adding additional arcs for every nondominated path between two nodes. The second approach consists in tackling the problem directly on a graph that mimics the road network, and that we call *road-network* graph. Theoretically, both approaches provide the same optimal solutions, that are at least as good as those obtained with the customer-based graph.

To the best of our knowledge, Garaix et al. [11] were the first to point out that using the customer-based graph leads to losing optimality when several attributes are defined on road segments. They investigated this for a dial-a-ride problem. The authors proposed to consider all alternative routes using a multigraph structure. They develop a branch-and-price procedure compatible with this multigraph. Later, Ben Ticha et al. [20] examined more extensively the same issue for the VRPTW, again with a branch-and-price algorithm. Their experimental analysis, conducted on several types of instances, confirm the gains achieved with the multigraph, compared to two customer-based graphs: a min-cost-path-based graph and a min-time-path-based graph.

Recently, Letchford et al. [15] revisited the branch-and-price approach presented by Garaix et al. [11]. They suggest that it could be more “natural” and more efficient to tackle the problem with a graph that mimics the road network. They explain how it would impact both the pricing problem and branching rules. However, they only explore the pricing problem. They demonstrate their approach for a multiple traveling salesman problem with time windows (m-TSPTW). The pricing problem is addressed with a dynamic programming algorithm adapted to the road-network graph. In the computational experiments, the efficiency of their method is compared with the approach of Garaix et al. [11]. Results confirm and illustrate the efficiency of pricing routines, applied in the road-network graph.

Although the results obtained by Letchford et al. [15] are insightful, their conclusions can hardly be generalized for many reasons. First, the problem that they consider, the m-TSPTW, involves a single resource in the pricing problem (namely, time), which has significant impacts on algorithm efficiency. Secondly, their experiments are based on instances with relatively high densities of customers: two sets of instances are considered with densities equal to 33% and 66% respectively. Real life applications are defined on large scale road networks in which a few number of nodes are associated with customer locations. Third, they only investigate the nonelementary-route version of the pricing problem (a route is said nonelementary when the service can be performed several times at customers). The case with only elementary routes is not considered. Finally, they only solve the pricing problem and leave the branching scheme for future researches. By the way, it is worth noting that, when nonelementary routes are allowed, lower bounds at the root node of the branch-and-price tree with the road-network graph and multigraph approaches are not guaranteed to be the same. It makes any comparison between the two approaches difficult if branching is not carried out.

This first study by Letchford et al. [15] thus leaves two important open questions: How can a complete branch-and-price scheme be developed based on the road-network graph? Would this be consistently more efficient than an equivalent branch-and-price scheme applied to the multigraph?

In this paper, we propose to follow the direction initiated by Letchford et al. [15] and aim to answer these questions. We indeed believe that it is crucial to better understand the pros and cons of these two models. Our main objectives are thus to: (1) investigate more in depth the relative efficiency of the two graphs, and (2) develop a complete branch-and-price solution methodology for the case of the road-network graph.

We consider as our test-bed problem the VRPTW, because it is probably the most studied VRP with two attributes (travel time and cost), and also because it admits the m-TSPTW (considered by Letchford et al. [15]) as special case. As we include road-network information in the VRPTW, we follow [2] and call it VRPTW with road-network information (VRPTW_{RN}). We develop a complete branch-and-price algorithm based on the pricing routines presented in Letchford et al. [15] and with an innovative methodology to manage the branching scheme. For the sake of comparison, we use the branch-and-price scheme developed in Ticha et al. [20] for the solution of the VRPTW_{RN} with a multigraph.

We propose an extensive computational study with three types of instances: instances generated by Letchford et al. [15]; a large set of instances similar to those of Letchford et al. [15]; instances derived from a real road network. On all these benchmark sets, we propose an empirical analysis of the impact of different factors (allowing or not nonelementary routes, capacity constraints, customer density, etc.) on the relative efficiency of the two branch-and-price algorithms.

The remainder of the paper is organized as follows. In Sections 2 and 3, we describe the branch-and-price algorithms for the road-network graph and multigraph, respectively. In Section 4, we report the computational results and we analyze the efficiency of the two models.

2 | BRANCH-AND-PRICE ALGORITHM FOR THE VRPTW ON THE ROAD-NETWORK GRAPH

The VRPTW_{RN} is defined as follows. Let $G = (V, A)$ be a directed graph that represents the road network. Arcs in A represent road segments, and nodes in V the extremities of these segments. With each arc $(i, j) \in A$ is associated a travel time t_{ij} and a

travel cost c_{ij} . Let node 0 represents the depot location. Let $C \subset V \setminus \{0\}$ represents the set of customers. With each customer $i \in C$ is associated a demand d_i , a time window $[e_i, l_i]$ and a service time s_i . A homogeneous fleet of K vehicles with a loading capacity Q is given. The aim is to find a set of routes of minimal total cost, starting and ending at the depot, and that serves each customer exactly once.

As for the standard VRPTW, the branch-and-price scheme for the $VRPTW_{RN}$ is based on the following set covering formulation:

$$\min \sum_{r \in \Omega} c_r x_r \quad (1)$$

$$s.t \quad \sum_{r \in \Omega} a_{i,r} x_r \geq 1 \quad \forall i \in C \quad (2)$$

$$\sum_{r \in \Omega} x_r \leq K \quad (3)$$

$$x_r \in \{0, 1\} \quad \forall r \in \Omega \quad (4)$$

In this context, Ω represents the set of feasible vehicle routes in the road-network graph, c_r represents the cost of route $r \in \Omega$ and $a_{i,r}$ is a binary parameter equal to 1 if and only if customer $i \in C$ is served along route r . Binary decision variables x_r take value 1 if route r is selected in the solution, 0 otherwise.

Due to the exponentially growing size of Ω , the optimal solution for the LP relaxation of (1) to (4), the so-called master problem (MP), cannot be computed using a standard linear programming procedure (e.g., the simplex method). This issue is handled using a column generation technique. Column generation is embedded into a branch-and-bound framework to obtain the optimal integer solution of (1) to (4).

In the column generation procedure, only a subset of columns $\Omega_1 \subset \Omega$ is considered and a restriction of MP to Ω_1 is solved at each iteration. Ω_1 consists of all the columns generated at previous iterations and is iteratively enlarged by solving the pricing problem. The pricing problem aims at finding new routes with negative reduced costs, that is, routes that offer better ways to visit customers. For a detailed description of the column generation algorithm, the reader is referred to [9].

2.1 | Pricing problem

Let λ_i , $i \in C$, be the dual variables associated with constraints (2) and λ_0 be the dual variable associated with constraint (3). The reduced cost of a route r is given by:

$$\hat{c}_r = c_r - \sum_{i \in C} a_{i,r} \lambda_i - \lambda_0 \quad (5)$$

The purpose of the pricing problem is to generate routes $r \in \Omega \setminus \Omega_1$ with $\hat{c}_r < 0$. Note that in the standard version of the VRPTW, the pricing problem is equivalent to an elementary shortest path problem with resource constraints (ESPPRC). This becomes clear by replacing c_r and $a_{i,r}$ in (5), by $\sum_{(i,j) \in A} b_{ijr} c_{ij}$ and $\sum_{j \in V \setminus \{i\}} b_{ijr}$ respectively, with $b_{ijr} = 1$ if route r traverses arc (i, j) and 0 otherwise. Using this notation, the reduced cost of route r can be expressed as:

$$\hat{c}_r = \sum_{(i,j) \in A} b_{ijr} (c_{ij} - \lambda_i) \quad (6)$$

Thus, the pricing problem is equivalent to the problem of finding: elementary paths starting and ending at the depot 0, satisfying time windows and capacity constraints and having a negative cost, with arc costs equal to $c_{ij} - \lambda_i$. This problem can efficiently be addressed with a dynamic programming approach [10, 17]. Desrochers et al. [8] show that the ESPPRC is NP-hard in the strong sense, however it can be solved in a pseudo-polynomial time when the elementary condition is relaxed. In this case, the pricing problem is reduced to a shortest path problem with resource constraints (SPPRC). Desrochers et al. [8] notice that, with the SPPRC, the enlargement of the set of feasible routes Ω does not affect the validity of the set covering formulation and of the branch-and-price scheme, but, it weakens significantly the lower bound. Also, slight modifications must be made to the branch-and-price framework: b_{ijr} must represent the number of times arc (i, j) is traversed along route r and $a_{i,r}$ the number of times customer i is served in r .

With the $VRPTW_{RN}$ and the road-network graph setting, two main issues arise. First, a customer node can be visited more than once in a route (but it is not necessarily served at each visit). Second, an arc can be traversed several times along a route. Thus, the pricing problem cannot be reduced to an ESPPRC. In this case, a route r is called elementary if every customer in r is served only once otherwise it is called nonelementary. To handle the road-network graph setting, the branch-and-price framework is modified as follows: b_{ijr} represents the number of times arc (i, j) is traversed along route r and $a_{i,r} = 1$ if customer i is served in r , 0 otherwise.

To solve the pricing problem, we adapt the algorithm proposed by Feillet et al. [10]. The solution procedure is based on a modified label correcting algorithm. A label represents a partial route from the depot node 0 to a node $v \in V$. It is defined using the following information, $L = (v, t, c, q, S)$ with:

- v is the last node visited in the partial route represented by L ;
- t represents the arrival time at v . When v is a served customer, t includes waiting and service times;
- c represents the reduced cost of the partial route represented by L ;
- q is the total demand of served customers along the partial route represented by L ;
- S represents the set of served customers along the partial route represented by L . S contains also the unreachable customers, that is, customers that cannot be served along the partial route represented by L without violating time or capacity constraints.

Algorithm 1 Pricing procedure in the road-network graph

```

1:  $L = (0, 0, 0, 0, \emptyset)$ 
2:  $Labels[0].add(L)$ 
3: for all  $i \in V$  do
4:    $Labels[i] \leftarrow \emptyset$ 
5: end for
6:  $W = \{0\}$ 
7: while  $W \neq \emptyset$  do
8:    $u = W.extract()$ 
9:   for all  $L = (u, t, c, q, S) \in Labels[u]$  do
10:    if  $L$  is not processed then
11:      for all  $(u, v) \in A$  do
12:        if  $v \in C \cup \{0\}$  and  $v \notin S$  then
13:          if  $t + t_{uv} \leq l_v$  and  $q + q_v \leq Q$  then
14:             $L' = (v, \max\{e_v, t + t_{uv}\} + s_v, c + c_{uv} - \lambda_v, q + q_v, S \cup \{v\})$ 
15:             $L'.updateUnreachableNodes()$ 
16:            if  $Labels[v].insert(L')$  then
17:               $W \leftarrow W \cup \{v\}$ 
18:            end if
19:          end if
20:        end if
21:         $L'' = (v, t + t_{uv}, c + c_{uv}, q, S)$ 
22:         $L''.updateUnreachableNodes()$ 
23:        if  $Labels[v].insert(L'')$  then
24:           $W \leftarrow W \cup \{v\}$ 
25:        end if
26:      end for
27:    end if
28:  end for
29: end while
30: return  $Labels[0]$ 

```

The developed algorithm is based on an exhaustive enumeration in which, for every label, all feasible extensions are performed. Once all labels are processed the algorithm terminates and all routes with negative reduced costs are constructed. Using this search strategy, the number of generated labels can be very large. To reduce this number, a dominance check is used. The dominance rule is defined as follows:

Definition 1. A label $L_1 = (v, t_1, c_1, q_1, S_1)$ dominates a label $L_2 = (v, t_2, c_2, q_2, S_2)$ if $t_1 \leq t_2$, $c_1 \leq c_2$, $q_1 \leq q_2$ and $S_1 \subseteq S_2$.

The general scheme of the pricing algorithm is described in Algorithm 1 where $Labels[v]$ is the list of labels whose last visited node is v and W is the list of active nodes, that is, at which there is a nonprocessed label. $Labels[v].insert(L)$ is a procedure that updates labels in $Labels[v]$ and keeps only nondominated labels. This procedure returns ‘False’ if L is dominated and has not been inserted. $W.extract()$ is a procedure that extracts a node from W . $L.updateUnreachableNodes$ is a procedure that updates the set of unreachable nodes in label L . In Algorithm 1, when the destination node is a customer $v \in C$, every label is extended

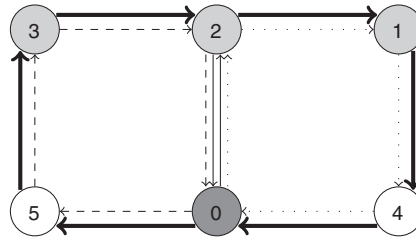


FIGURE 1 Example of a fractional solution supported by an integer arc flow

twice: in the first extension the customer v is served, if it is possible regarding time and capacity constraints, and in the second extension v is visited without being served.

Note that this pricing procedure can be easily adapted to allow for nonelementary routes. To do this, we do not need to check if customer v was already served at each extension of a label along an arc $(u, v) \in A$ (line 12 of Algorithm 1). Also, the comparison of sets of unreachable nodes is not considered for the dominance checks.

2.2 | Branching scheme

The branching rule is one of the important components of the branch-and-price scheme. It iteratively splits the solution space to restrict the search and tighten the bounds. It is important to make sure that the added constraints are compatible with the column generation procedure.

For VRP, the standard branching rule relies on the property that in a feasible solution each arc is at most traversed by one vehicle (see [13] for more details). Let ϕ_{ij} denotes the flow on arc (i, j) , that is, $\phi_{ij} = \sum_{r \in \Omega_1} b_{ijr} x_{ijr}$. The standard branching rule consists in selecting an arc (i, j) such that $0 < \phi_{ij} < 1$ then, deriving two branches: in the first branch, the use of arc (i, j) in the solution is forbidden, and in the second branch, arc (i, j) is enforced in the solution. This branching rule is very easy to manage in both the pricing and the master problems.

Unfortunately, the standard branching rule cannot be used with the road-network graph since an arc can be traversed several times and by several vehicles. In our implementation, we propose to use a branching rule that works as follows:

- Select an arc $(i, j) \in A$ with fractional flow $\phi_{ij} > 0$
- Derive two branches:
 - In the first branch, the upper limit on flow on arc (i, j) is fixed to $\lfloor \phi_{ij} \rfloor$
 - In the second branch, the lower limit on flow on arc (i, j) is fixed to $\lfloor \phi_{ij} \rfloor + 1$

This branching rule is very easily managed in the column generation scheme. Constraints

$$\sum_{r \in \Omega_1} b_{ijr} x_r \leq \lfloor \phi_{ij} \rfloor \quad (7)$$

and

$$\sum_{r \in \Omega_1} b_{ijr} x_r \geq \lfloor \phi_{ij} \rfloor + 1 \quad (8)$$

are added to MP in the first and second branches, respectively. At the pricing problem level, λ_{ij}^{up} (resp., λ_{ij}^{down}) is subtracted to the cost of arc (i, j) , where $\lambda_{ij}^{down} \leq 0$ is the dual variable associated with constraint (7) (resp. $\lambda_{ij}^{up} \geq 0$ is the dual variable associated with constraint (8)).

Note that, it is shown (see [1]) that for the standard VRPTW, an integer flow corresponds to an integer routing solution. Thus, when no arc with a fractional flow is found, the obtained solution is optimal for the associated branch. With the road-network graph, this property does not hold. A fractional routing solution could be supported by an integer flow. To illustrate this, let us consider the example provided in Figure 1 (note that this example is somewhat artificial because deriving a small example is not easy).

In this example, the depot is located at node 0, and customers are located at nodes 1, 2 and 3. Other nodes represent road junctions. Suppose that we are given a fleet with enough vehicles and let us consider the following routes (assumed to be feasible regarding time windows and capacity constraints):

- $r_1 = (0, 2, 0)$, represented with thin lines;
- $r_2 = (0, 5, 3, 2, 0)$, represented with dashed lines;

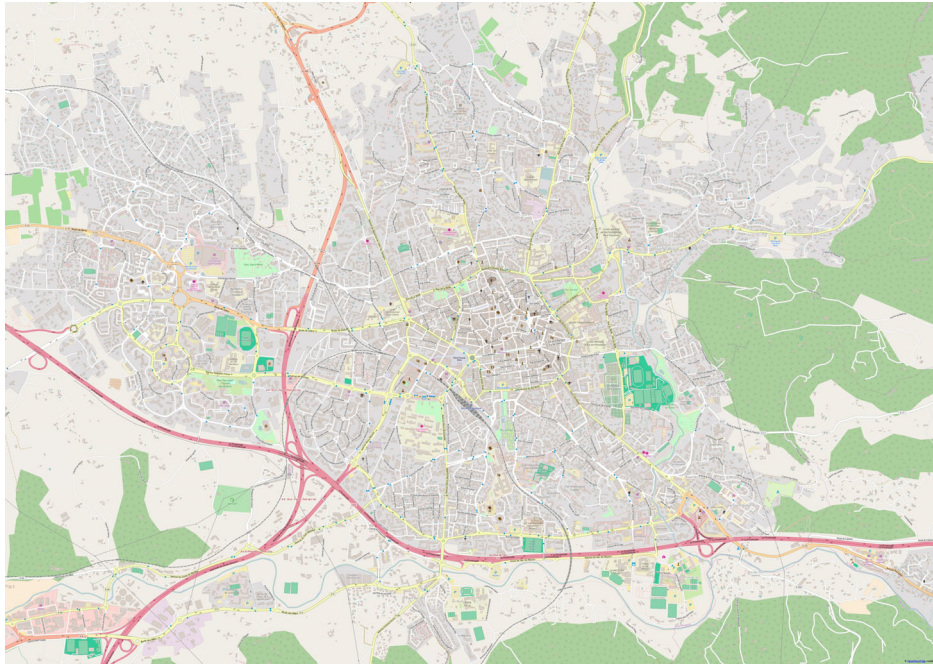


FIGURE 2 Road network of the central urban area of Aix-en-Provence [Colour figure can be viewed at wileyonlinelibrary.com]

- $r_3 = (0, 2, 1, 4, 0)$, represented with dotted lines;
- $r_4 = (0, 5, 3, 2, 1, 4, 0)$, represented with thick lines.

The solution given by $x_{r_1} = x_{r_2} = x_{r_3} = x_{r_4} = 0.5$ is fractional, feasible for MP and defines an integer flow.

To tackle this difficulty, we propose a procedure based on a specific branching scheme. Suppose that at a certain node of the search tree, the obtained solution $\tilde{x} = (\tilde{x}_r)_{r \in \Omega_1}$ is fractional and the associated flow is integer, that is, $\sum_{r \in \Omega_1} b_{ijr} \tilde{x}_r$ is integer for all arcs $(i, j) \in A$. Let $\tilde{G} = (V, \tilde{A})$ be the graph induced by this solution, that is, $\tilde{A} = \{(i, j) \in A : \sum_{r \in \Omega_1} b_{ijr} \tilde{x}_r > 0\}$. The proposed procedure derives two branches:

- In the first branch, we seek for a feasible solution defined on graph \tilde{G} . For this aim, we apply Algorithm 1 (in which the dominance rule is deactivated in the insertion procedure) to enumerate the complete set of feasible routes that may exist in \tilde{G} , denoted by $\tilde{\Omega} \subset \Omega$. Then, a set covering problem is solved based on set $\tilde{\Omega}$ with an IP solver;
- In the second branch, we enforce the use of at least one arc that is not used by solution \tilde{x} . To do this, constraint (9) is added to the master problem.

$$\sum_{(i,j) \in A \setminus \tilde{A}} \sum_{r \in \Omega_1} b_{ijr} x_r \geq 1 \quad (9)$$

At the pricing problem level, a dual variable $\tilde{\lambda}_a \geq 0$ is subtracted from the cost of each arc a appearing in constraint (9).

It is worth mentioning that Bode and Irnich [5] propose a branching scheme for the Capacitated Arc Routing Problem (CARP), whose graph is also a road-network graph. Their proposed branching scheme is based on three levels of decisions: (1) branching on node degrees, (2) branching on edge flows, and (3) branching on *followers and nonfollowers*. The first level consists in branching once a node with a noneven (fractional or odd) degree is found. The second level is similar to the one described by constraints (7) and (8). These two branching rules are mainly used to improve the quality of the lower bound and do not guarantee the integrality of the solution. The *followers and nonfollowers* branching rule consists in deciding whether two required edges are serviced consecutively in the same route or not. Bode and Irnich [5] show that this branching rule can ensure the integrality of the routing solution for the CARP. Unfortunately, this branching rule is not suitable for our problem. Indeed, several alternative paths can exist between each pair of customer nodes. To address the *follower* constraint between two nodes i and j , one would need to compute all nondominated paths linking i to j , which involves solving a (NP-hard) multi-objective shortest path problem [18]. In addition, the number of efficient paths grows exponentially with the number of nodes in the network, which makes the pricing problem computationally very expensive.

Conversely, our approach would probably not be practical for the CARP. Indeed, we exploit the fact that in our context vehicles only traverse a very limited part of the network. Hence, generating all feasible routes in \tilde{G} makes sense. In the CARP, vehicles typically cover a large proportion of arcs and \tilde{G} is very similar, if not equal, to G .

3 | BRANCH-AND-PRICE ALGORITHM FOR THE VRPTW_{RN} ON A MULTIGRAPH

In this section, we briefly describe the branch-and-price scheme for the VRPTW_{RN} when a multigraph is introduced. A full description can be found in [20]. The multigraph $G^{MG} = (V^{MG}, A^{MG})$ is defined as follows. $V^{MG} = C \cup \{0\}$. A^{MG} contains parallel arcs between each pair of nodes: $A^{MG} = \cup_{i,j \in V^{MG}} A_{ij}^{MG}$, where $A_{ij}^{MG} = \{(ij)^p, p = 1, \dots, m_{ij}\}$ is the set of efficient paths linking customer locations i and j in the road network. Each arc $(i, j)^p \in A^{MG}$ is given a travel time t_{ij}^p and a travel cost c_{ij}^p that represent respectively the time needed and the cost induced to travel from i to j using the path indexed by p .

With the multigraph, the Master Problem is the same as for the standard VRPTW. However, some modifications are needed for the column generation scheme. First, the pricing problem is an ESPPRC applied on multigraph G^{MG} , where we generate elementary routes that satisfy:

$$\sum_{(i,j) \in A^{MG}} \sum_{p=1}^{|A_{ij}|} \alpha_{ijp}^r (c_{ij}^p - \lambda_i) < 0 \quad (10)$$

with $\alpha_{ijp}^r = 1$ if route r uses arc $(i, j)^p$ and λ_i is the dual variable associated with constraint (2).

Algorithm 1 is adapted to handle the multigraph setting: a label at some node $i \in V^{MG}$ is extended along all arcs $(i, j)^p \in A^{MG}$ (instead of A in line 11 of Algorithm 1) and labels are only extended with a service or to the depot (lines 21-25 of Algorithm 1 are not considered).

The branching rule is like the standard one. If for any arc $(i, j)^p \in A^{MG}$ the flow is fractional, two branches are generated. In the first branch, the use of arc $(i, j)^p$ is forbidden and in the second branch arc $(i, j)^p$ is imposed in the solution.

4 | COMPUTATIONAL EXPERIMENTS

In this section, we present the computational experiments to evaluate the performance of the branch-and-price algorithm applied on the road-network graph, and to compare it with the branch-and-price on the multigraph. In Section 4.1, we present the datasets used in the experiments. In Section 4.2, we report the results. A discussion follows in Section 4.3.

Branch-and-price algorithms are implemented in the C++ programming language. Tests are run on an Intel CORE i5 2.6 GHz computer with 4GB of memory. We use ILOG CPLEX 12.6 as the linear programming solver for restricted master problems.

4.1 | Datasets

In our experiments, we use three sets of instances. The first set is provided by Letchford et al. [15]. The second set is generated using the procedure proposed in [15] (described below). The third set consists of instances derived from the real road network of the city of *Aix-in-Provence*, France. The three sets are respectively called LET, NEWLET, and AIX.

4.1.1 | LET and NEWLET instances

LET instances were generated by Letchford et al. [15] with the objective of simulating real-life road networks, using the following procedure:

1. Insert nodes at random positions in the Euclidean space;
2. Consider all possible arcs and insert new arcs sequentially (to represent road segments) on condition that the new inserted arc does not intersect with any other arc and has sufficiently large angles with other arcs at its endpoints;
3. Set arc costs to the Euclidean distance between arc endpoints.

Using this procedure, Letchford et al. [15] generated different sparse graphs of different sizes. In each graph, a node is randomly selected to be the depot and other nodes are given a probability p to be customers. For each sparse graph, Letchford et al. [16] generated three different sets of travel times with different levels of correlation with travel costs. These travel times are computed with formula $t_{ij} = v^*c_{ij} + \mu^*\gamma_{ij}^*\bar{c}$ where γ_{ij} is a random number in $[0, 1]$, $\bar{c} = \max_{(i,j) \in A} c_{ij}$, and parameters v, μ ,

$\mu \in [0, 1]$ define the correlation degree. Finally, Letchford et al. [15] associate with each graph two m-TSPTW instances: a first instance with wide time windows (WTW) and a second instance with narrow time windows (NTW). Time windows are defined such that a set of routes, constructed in a greedy way, are feasible. An integer service time in $\{1, 2\}$ is defined for each customer node.

In our experiments, we consider six of these LET instances with $|V|=200$ and $p=0.66$: two instances with noncorrelated travel times, two instances with weakly-correlated travel times and two instances with strongly-correlated travel times. In order to define VRPTW_{RN} instances, we introduce a vehicle capacity, set to 200, we consider a fleet with a large number of vehicles, and we assign a demand to each customer such that the routes defined by the time windows remain feasible.

Using the procedure described above, we generated 450 NEWLET instances. We consider the same values for $|V|$ and p as in [15]: $P=0.66$ for $|V| \in \{50, 100, 150, 200, 250\}$ and $P=0.33$ for $|V| \in \{100, 200, 300, 400, 500\}$. Considering the three correlation levels and the two types of time-windows, it gives a first subset of 60 instances. We also generated 30 additional instances for $|V|=250$, by varying customer density p : $p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. Finally the graph generation procedure is repeated five times to obtain five replications of these 90 instances.

4.1.2 | AIX instances

AIX instances are based on the real road network of the central urban area of *Aix-en-Provence* (a city-commune in the south of France). Spatial data are extracted from the OpenStreetMap^① database. It provides a road-network graph $G=(V, A)$ with $|V|=5437$ and $|A|=10\,181$. In the database, each arc represents a road segment and has known length and maximum allowed speed. These values are used to define travel costs c_{ij} and travel times t_{ij} . The network is depicted on Figure 2.

Using this graph, we generate 20 instances with $|C| \in \{5, 10, 25, 50\}$ (five instances for each $|C|$). For each instance, depot and customer locations are randomly selected. Problem characteristics (time windows, customer demands, service times and vehicle capacity) are defined in the same way as for NEWLET instances.

4.2 | Result tables

To complete the results presented by Letchford et al. [15] and to derive comprehensive conclusions, we propose the following experimental plan, composed of four parts. In the four parts, we compare the solutions obtained with the branch-and-price algorithms designed for the road-network graph and for the multigraph:

1. We consider the m-TSPTW and only compute the LP relaxation, with nonelementary routes allowed;
2. We consider the m-TSPTW and compute the LP relaxation with elementary routes only;
3. We consider the VRPTW_{RN} and compute the LP relaxation with elementary routes only;
4. We consider the VRPTW_{RN} and apply the complete branch-and-price with elementary routes only.

Note that in the case of the m-TSPTW, the demand resource is removed from label definition.

In the first three parts of our experimental plan, we use LET and NEWLET instances only. AIX instances are added for the last part, when integer solutions are searched. For all experiments, computing times are expressed in seconds and limited to 7200 seconds.

Multigraphs are generated with the method described in [3] for all instances. We do not include multigraph construction time in the reported computing times.

4.3 | Part 1: m-TSPTW, LP relaxation, nonelementary routes

Results are reported in Tables 1 and 2. In Table 1, results for LET and most NEWLET instances are presented. Table 2 concerns NEWLET instances with $|V|=250$ and varying customer density.

In these tables, each line gives average results on five instances. The first three columns describe the instance characteristics, with average values reported for $|A|$ and $|A^{MG}|$. NC, WC and SC respectively stand for noncorrelated, weakly-correlated and strongly-correlated travel times and costs. In Table 1, triplet $(|V|, |A|, |C|)$ is marked with an asterisk for LET instances, in order to better distinguish them. Following columns are repeated for the two types of time windows: narrow and wide.

Column *Gap* gives the average gap between the lower bounds obtained with the multigraph (LB_{MG}) and the road-network graph (LB_{RN}):

$$Gap(\%) = \frac{LB_{RN} - LB_{MG}}{LB_{MG}} \times 100 \quad (11)$$

^①OpenStreetMap is a collaborative project which creates and distributes freely available geo-spatial data. www.openstreetmap.org/

Note that, this gap is because, with the road-network graph and when nonelementary routes are allowed, a customer can be served several times consecutively (with intermediate crossroads), while in the multigraph one has to serve at least one intermediate customer.

Columns $CPU(s)$ provide average computing times for the multigraph (MG) and the road-network graph (RN). Additional columns report the minimal, average and maximal ratios between these computing times. Minimal and maximal ratios are not given for the LET instances if not replicated five times.

Tables 1 and 2 show that the lower bounds obtained with the multigraph are better in most cases. Average gaps can reach very large values as -59.3% for strongly correlated instances with $|V|=250$, $|C|=25$ and WTW. From Table 1, we also observe that gaps generally increase with the correlation level. We also see that gaps are larger when time windows are wide. For NEWLET instances with $|V|=250$, Table 2, we see that the average gap decreases with the number of customers.

From Tables 1 and 2, computing times are larger for road-network graphs. Ratio CPU_{MG}/CPU_{RN} is less than 1 for the majority of instances: over the 456 instances, the ratio is greater than 1 for only 27 instances. The average ratio is larger when time windows are narrow, and it increases when the correlation between travel times and costs decreases. Another observation is that

TABLE 1 LP relaxation with nonelementary routes for the m-TSPTW

(V , A , C)	Corr	$ A^{MG} $	Narrow time windows						Wide time windows					
			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		
			Gap (%)	MG	RN	Min	Avg	Max	Gap (%)	MG	RN	Min	Avg	Max
(50,135,33)	NC	2362	-0.3	0.3	0.4	0.56	0.83	1.22	-3.9	0.5	0.9	0.39	0.49	0.60
	WC	1864	-0.7	0.4	0.5	0.43	0.67	0.96	-8.2	0.5	1.3	0.26	0.42	0.52
	SC	1266	-2.2	0.3	0.4	0.45	0.65	0.91	-10.9	0.5	1.3	0.33	0.41	0.48
(100,278,66)	NC	11 795	-0.1	1.4	1.4	0.72	1.18	2.08	-3.4	2.2	4.2	0.37	0.55	0.71
	WC	9581	0.0	1.6	1.9	0.63	0.88	1.16	-5.0	2.9	6.4	0.34	0.55	0.75
	SC	5265	-2.2	1.8	2.3	0.51	0.79	1.03	-7.2	3.7	11.2	0.29	0.34	0.42
(150,429,100)	NC	32 346	-0.1	4.5	5.2	0.73	0.95	1.44	-3.7	9.3	19.7	0.38	0.55	0.80
	WC	25 561	-0.1	4.5	5.3	0.67	0.88	1.11	-5.2	10.4	19.5	0.41	0.55	0.72
	SC	13 193	-2.6	4.8	5.5	0.56	0.87	1.23	-6.2	11.3	27.8	0.35	0.41	0.44
(200,574,133)	NC	68 979	-0.4	10.9	11.7	0.81	0.95	1.12	-3.7	28.2	56.5	0.36	0.57	0.97
	WC	52 742	-0.1	12.1	13.5	0.72	0.94	1.32	-3.9	24.9	45.9	0.40	0.58	0.89
	SC	23 798	-3.5	12.0	12.2	0.51	1.05	1.79	-6.5	25.7	75.0	0.25	0.39	0.67
(200,582,133) ^a	NC	50 368	-2.1	45.1	34.6			1.30	-4.3	54.3	43.7			1.24
	WC	38 748	-1.3	16.2	17.9			0.91	-2.7	44.0	29.7			1.48
	SC	24 758	-2.0	13.3	16.4			0.81	-2.3	31.2	27.8			1.12
(250,714,166)	NC	116 300	0.0	23.9	26.5	0.62	1.06	1.56	-2.4	44.1	62.4	0.55	0.79	1.02
	WC	92 500	-0.1	25.0	22.2	0.78	1.24	1.93	-3.9	44.4	71.8	0.48	0.65	0.85
	SC	37 254	-2.1	14.5	19.9	0.35	0.80	1.21	-5.4	50.1	135.3	0.32	0.39	0.55
(100,278,33)	NC	2994	-0.3	0.2	0.5	0.29	0.41	0.50	-5.1	0.3	1.1	0.21	0.29	0.53
	WC	2400	-1.0	0.2	0.5	0.36	0.44	0.56	-12.3	0.3	1.7	0.14	0.22	0.35
	SC	1315	-3.8	0.3	1.1	0.20	0.29	0.41	-19.2	0.5	2.9	0.14	0.17	0.20
(200,574,66)	NC	16 954	-0.1	1.7	4.6	0.22	0.41	0.64	-6.3	2.2	12.5	0.16	0.18	0.24
	WC	13 076	-0.1	1.4	4.2	0.23	0.35	0.44	-11.2	2.7	13.2	0.09	0.22	0.30
	SC	5806	-4.1	1.2	5.3	0.19	0.23	0.29	-13.8	2.2	21.3	0.07	0.11	0.15
(300,869,100)	NC	50 418	-0.1	6.8	11.9	0.43	0.57	0.71	-4.1	7.8	24.6	0.24	0.33	0.43
	WC	39 084	-0.3	4.5	9.9	0.35	0.47	0.59	-7.6	7.7	27.9	0.21	0.30	0.42
	SC	14 545	-2.5	3.7	12.8	0.22	0.31	0.45	-13.2	7.5	68.9	0.09	0.12	0.18
(400,1165,133)	NC	103 356	0.0	13.4	28.9	0.34	0.53	0.69	-7.3	25.1	80.0	0.25	0.35	0.52
	WC	77 037	-0.1	15.0	26.6	0.45	0.60	0.93	-9.8	23.7	100.4	0.16	0.27	0.34
	SC	26 583	-4.2	7.8	17.4	0.32	0.44	0.64	-12.1	21.6	113.9	0.15	0.20	0.25
(500,1458,166)	NC	196 267	-0.1	38.8	56.7	0.61	0.73	1.06	-4.7	44.3	134.9	0.26	0.37	0.59
	WC	145 782	-0.1	30.1	75.1	0.31	0.55	0.97	-9.0	46.3	174.7	0.18	0.30	0.41
	SC	44 751	-3.2	20.5	62.0	0.19	0.40	0.58	-12.9	32.8	271.4	0.09	0.13	0.17

Abbreviation: TSPTW = traveling salesman problem with time windows.
LET and NEWLET instances.

TABLE 2 LP relaxation with nonelementary routes for the m-TSPTW (NEWLET instances with $|V| = 250$)

C	Corr	$ A^{MG} $	Narrow time windows						Wide time windows					
			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		
			Gap (%)	MG	RN	Min	Avg	Max	Gap (%)	MG	RN	Min	Avg	Max
25	NC	2553	0.0	0.1	1.1	0.09	0.14	0.19	-12.8	0.2	2.5	0.05	0.07	0.09
	WC	2123	0.0	0.1	1.0	0.11	0.15	0.19	-49.3	0.2	3.2	0.03	0.05	0.06
	SC	878	-17.7	0.1	1.4	0.08	0.09	0.10	-59.3	0.1	5.9	0.02	0.03	0.05
50	NC	10704	-0.5	0.8	3.3	0.09	0.23	0.34	-8.0	1.0	7.4	0.11	0.14	0.19
	WC	8697	-0.1	0.7	3.2	0.11	0.25	0.37	-15.2	1.3	11.7	0.09	0.13	0.20
	SC	3478	-5.5	0.6	3.7	0.13	0.18	0.30	-21.5	1.3	16.0	0.05	0.08	0.12
75	NC	24267	-0.2	2.4	5.9	0.22	0.41	0.64	-6.2	3.0	12.5	0.16	0.18	0.24
	WC	19333	0.0	1.8	5.5	0.23	0.35	0.44	-9.9	3.5	21.8	0.09	0.22	0.30
	SC	7815	-3.4	1.7	7.1	0.19	0.23	0.29	-17.5	3.9	41.8	0.07	0.11	0.15
100	NC	42367	-0.3	5.8	9.3	0.34	0.43	0.56	-6.4	8.3	23.6	0.21	0.32	0.48
	WC	34163	0.0	4.9	9.5	0.23	0.34	0.53	-8.6	9.0	33.7	0.15	0.21	0.30
	SC	13716	-3.0	3.1	9.4	0.20	0.37	0.55	-10.9	9.9	63.3	0.15	0.18	0.22
125	NC	66248	-0.2	10.5	13.3	0.47	0.62	0.84	-4.7	15.8	36.4	0.18	0.35	0.48
	WC	52534	-0.7	9.9	12.9	0.46	0.59	0.93	-6.0	18.8	45.0	0.20	0.31	0.46
	SC	21336	-1.5	7.4	12.9	0.39	0.63	0.89	-9.0	18.5	84.4	0.15	0.24	0.34
166	NC	116300	0.0	23.9	26.5	0.81	0.95	1.12	-2.6	44.1	62.4	0.36	0.57	0.97
	WC	92500	-0.1	25.0	22.2	0.72	0.94	1.32	-4.2	44.4	71.8	0.40	0.58	0.89
	SC	37254	-2.2	14.5	19.9	0.51	1.05	1.79	-5.9	50.1	135.3	0.25	0.39	0.67

Abbreviation: TSPTW = traveling salesman problem with time windows.

the ratio increases with the number of customers. Figure 3 depicts the evolution of CPU_{MG}/CPU_{RN} with the number of customers for instances with $|V| = 250$ and for both types of time windows. The figure tends to show that applying branch-and-price on the road-network graph might only be beneficial for instances with NTW and high density of customers.

4.3.1 | Part 2: m-TSPTW, LP relaxation, elementary routes

Results are reported in Tables 3 and 4. When only elementary routes are allowed, the LP bound is the same whether the road-network graph or the multigraph is used. For that reason, the *Gap* column is removed. Apart from that, the structure is the same as in Tables 1 and 2.

The first observation with Tables 3 and 4 is that computing times increase when elementary-path restrictions are introduced. The computing time increase is however much stronger for the road-network graph. Indeed, ratios CPU_{MG}/CPU_{RN} tend to become very small, especially for large instances and NTW. The impact of the time window width is more limited for road-network graphs. It is not surprising since time constraints only apply at a small subset of nodes in these graphs, while they apply at all nodes in multigraphs.

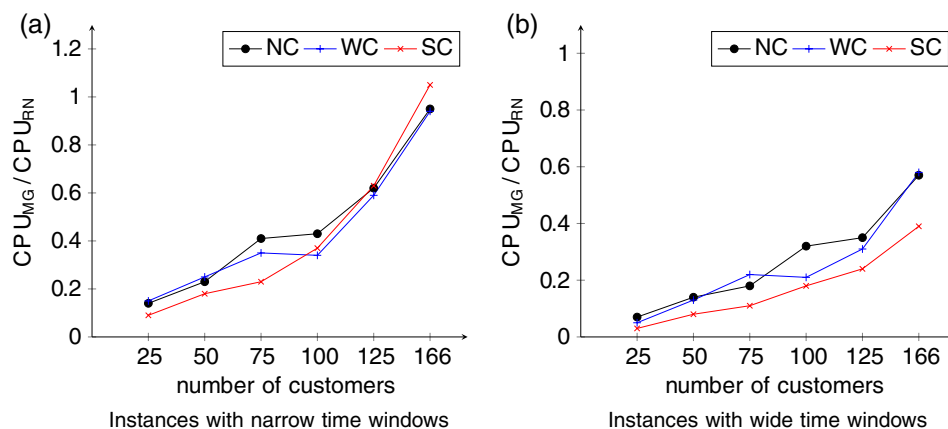


FIGURE 3 Evolution of the ratio $\frac{CPU_{MG}}{CPU_{RN}}$ with the number of customers for instances with $n = 250$ [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 3 LP relaxation with elementary routes for the m-TSPTW

(V , A , C)	Corr	A ^{MG}	Narrow time windows					Wide time windows				
			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$	
			MG	RN	Min	Avg	Max	MG	RN	Min	Avg	Max
(50,135,33)	NC	2362	0.2	0.9	0.16	0.22	0.33	0.4	1.5	0.18	0.29	0.41
	WC	1864	0.2	1.7	0.09	0.12	0.21	0.6	2.1	0.21	0.28	0.47
	SC	1266	0.1	1.1	0.06	0.13	0.21	0.5	2.1	0.13	0.24	0.60
(100,278,66)	NC	11 795	0.8	5.8	0.08	0.14	0.19	2.6	7.7	0.23	0.34	0.47
	WC	9581	0.9	6.5	0.10	0.14	0.21	5.6	10.9	0.28	0.52	0.76
	SC	5265	0.7	9.4	0.04	0.07	0.21	7.4	14.6	0.37	0.51	0.81
(150,429,100)	NC	32 346	3.4	23.2	0.09	0.15	0.24	24.4	41.8	0.43	0.58	0.69
	WC	25 561	3.5	29.1	0.10	0.12	0.14	26.8	42.4	0.42	0.63	0.83
	SC	13 193	2.8	42.7	0.03	0.07	0.12	38.2	56.3	0.59	0.68	0.76
(200,574,133)	NC	68 979	9.6	59.1	0.11	0.16	0.45	75.0	99.5	0.43	0.75	1.02
	WC	52 742	7.5	74.2	0.08	0.10	0.22	85.5	96.0	0.40	0.89	1.60
	SC	23 798	7.1	95.8	0.04	0.07	0.38	123.5	140.7	0.43	0.88	1.36
(200,582,133) ^d	NC	50 368	75.8	175.0			0.43	456.5	265.1			1.72
	WC	38 748	25.5	129.6			0.20	239.6	194.6			1.23
	SC	24 758	19.7	101.6			0.19	199.2	132.7			1.50
(250,714,166)	NC	116 300	20.6	186.4	0.05	0.11	0.22	122.8	188.8	0.19	0.65	1.25
	WC	92 500	20.2	138.5	0.13	0.15	0.19	165.5	211.8	0.24	0.78	1.24
	SC	37 254	13.2	134.6	0.06	0.10	0.16	268.8	278.9	0.61	0.96	1.11
(100,278,33)	NC	2993.6	0.2	1.8	0.07	0.10	0.16	0.3	2.3	0.10	0.12	0.17
	WC	2399.6	0.2	1.9	0.07	0.10	0.19	0.3	3.1	0.09	0.10	0.11
	SC	1314.6	0.2	5.2	0.01	0.04	0.09	0.4	4.1	0.07	0.10	0.17
(200,574,66)	NC	16 954	0.5	12.2	0.04	0.04	0.37	1.0	17.4	0.04	0.06	0.77
	WC	13 076	0.3	15.9	0.01	0.02	0.51	1.3	18.4	0.03	0.07	0.69
	SC	5806	0.3	19.4	0.01	0.02	0.42	1.6	20.7	0.04	0.08	0.70
(300,869,100)	NC	50 418	3.4	91.5	0.03	0.04	0.05	8.7	94.1	0.05	0.09	0.20
	WC	39 084	2.8	67.7	0.02	0.04	0.06	11.7	100.6	0.05	0.12	0.24
	SC	14 545	1.9	84.2	0.02	0.02	0.03	11.3	145.2	0.06	0.08	0.13
(400,1165,133)	NC	103 356	11.1	169.7	0.05	0.07	0.08	38.6	239.7	0.11	0.16	0.29
	WC	77 037	10.2	196.8	0.04	0.05	0.11	54.4	296.7	0.08	0.18	0.30
	SC	26 583	5.9	144.4	0.03	0.04	0.05	58.0	267.6	0.12	0.22	0.34
(500,1458,166)	NC	196 267	28.0	391.5	0.06	0.07	0.09	84.9	415.4	0.13	0.20	0.31
	WC	145 782	25.1	417.5	0.06	0.06	0.07	112.9	656.0	0.12	0.17	0.20
	SC	44 751	14.8	407.2	0.03	0.04	0.05	138.2	499.5	0.18	0.28	0.42

Abbreviation: TSPTW = traveling salesman problem with time windows. LET and NEWLET instances.

Table 4 also shows that increasing the number of customers for a fixed network size $|V|$ globally increases the ratio CPU_{MG}/CPU_{RN} . This ratio is 0.02 on average for NC instances with $|C| = 25$ and reaches 1.25 for NC instances with $|C| = 166$.

4.3.2 | Part 3: VRPTW_{RN}, LP relaxation, elementary routes

We now address the computation of the LP relaxation for the VRPTW_{RN}. Results are presented in Tables 5 and 6.

From these tables, it is observed that, when adding vehicle capacity, the average computing time required by column generation increases for both modeling approaches. This increase is more important for the road-network graph than for the multigraph. For example, regarding instances with NTW, CPU_{MG} is on average three times higher for the VRPTW than for the m-TSPTW and CPU_{RN} is on average six times higher. Consequently, the ratio CPU_{MG}/CPU_{RN} decreases for the VRPTW_{RN}, which indicates that the pricing algorithm is much faster with the multigraph. For all test-problem configurations, the ratio does not exceed 0.22 for NTW and 0.70 for WTW.

Table 6 shows that increasing the density of customers limits the difference between CPU_{MG} and CPU_{RN} . Seeing NC instances with NTW for example, the column generation procedure is 9.1 times faster when applied to the multigraph for $|C| = 166$ and 50 times faster for $|C| = 25$.

TABLE 4 LP relaxation with elementary routes for the m-TSPTW (NEWLET instances with $|V| = 250$)

C	Corr	$ A^{MG} $	Narrow time windows						Wide time windows					
			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$		
			MG	RN	Min	Avg	Max	MG	RN	Min	Avg	Max		
25	NC	2553	0.1	6.9	0.01	0.02	0.02	0.1	8.3	0.01	0.02	0.02		
	WC	2123	0.1	8.9	0.01	0.01	0.02	0.2	7.9	0.01	0.02	0.02		
	SC	878	0.1	8.8	0.01	0.01	0.02	0.1	10.2	0.01	0.01	0.02		
50	NC	10704	0.4	17.6	0.02	0.02	0.03	1.1	23.5	0.02	0.05	0.09		
	WC	8697	0.4	15.5	0.02	0.03	0.04	1.3	25.7	0.02	0.05	0.08		
	SC	3478	0.3	20.6	0.01	0.02	0.03	1.2	33.4	0.03	0.04	0.06		
75	NC	24267	1.6	33.4	0.03	0.05	0.07	3.6	40.9	0.05	0.10	0.16		
	WC	19333	1.3	33.6	0.04	0.04	0.04	4.1	44.0	0.07	0.09	0.11		
	SC	7815	1.1	44.9	0.01	0.02	0.03	7.3	69.6	0.04	0.11	0.17		
100	NC	42367	3.5	63.9	0.03	0.06	0.08	10.7	79.7	0.06	0.14	0.23		
	WC	34163	3.3	58.3	0.04	0.06	0.07	20.1	73.1	0.13	0.24	0.55		
	SC	13716	2.5	60.8	0.03	0.04	0.07	25.9	97.3	0.20	0.26	0.34		
125	NC	66248	7.7	87.2	0.06	0.09	0.15	28.1	87.0	0.16	0.31	0.52		
	WC	52534	7.3	101.3	0.05	0.08	0.10	52.2	127.2	0.24	0.41	0.53		
	SC	21336	4.9	95.0	0.03	0.06	0.08	72.5	159.4	0.28	0.50	0.84		
166	NC	116300	20.6	186.4	0.05	0.14	0.22	122.8	188.8	0.19	0.67	1.25		
	WC	92500	20.2	138.5	0.13	0.15	0.19	165.5	211.8	0.24	0.83	1.24		
	SC	37254	13.2	134.6	0.06	0.11	0.16	268.8	278.9	0.61	0.94	1.11		

Abbreviation: TSPTW = traveling salesman problem with time windows.

4.3.3 | Part 4: VRPTW_{RN}, branch-and-price, elementary routes

In Tables 7–9, we now solve the VRPTW_{RN} with the complete branch-and-price algorithm.

Tables 7 and 8 report results for LET and NEWLET instances. In these tables, columns *Solved* indicate the number of instances solved within the time limit (7200 seconds) using the multigraph or the road-network graph.

In these tables, we observe that, when time windows are narrow, 227 instances out of 228 are solved with the multigraph-based branch-and-price, against 207 for the road-network-based branch-and-price. When time windows are wide, 127 and 99 instances out of 228 are solved, respectively. This clearly shows the superiority of the multigraph model on these instances and with this algorithm. We also observe that average computing times increase for both approaches when extending the customer time windows.

Table 9 reports results for AIX instances. In these instances, customer density is much smaller. Even with $|C| = 50$, the density is less than 0.01. In our opinion, these low levels of density are much more representative of real-world situations than those of LET and NEWLET instances.

Columns Sav_{MC} and Sav_{MT} indicate the savings on solution cost achieved when using the multigraph or the road-network graph compared to two customer-based graphs: the min-cost-path customer-based graph (MC) and the min-time-path customer-based graph (MT), respectively. We introduce these columns to illustrate the interest of considering full road-network information. In this table, each line reports results for a single instance.

We see that all instances are solved very quickly using the multigraph. The large size of the road network does not complicate the solution with the multigraph. On the contrary, solution with the road-network graph is slow, even for very small instances with five customers. Two instances with 50 customers could not be solved in 2 hours.

Table 9 also shows that the solution quality is significantly improved when considering road-network information. The savings reach 17.6% against the min-cost graph and 17.4% against the min-time graph. These results confirm what have already been observed on similar instances in [20].

4.4 | Discussion

The computational study does not confirm that a road-network-graph-based branch-and-price scheme is more efficient than a multigraph-based branch-and-price, as previously stated by Letchford et al. [15]). Numerical results show that the efficiency depends on several factors.

TABLE 5 LP relaxation with elementary routes for the VRPTW

(V , A , C)	Corr	A ^{MG}	Narrow time windows					Wide time windows				
			CPU (s)		$\frac{CPU_{MG}}{CPU_{RN}}$			CPU (s)		$\frac{CPU_{MG}}{CPU_{RN}}$		
			MG	RN	Min	Avg	Max	MG	RN	Min	Avg	Max
(50,135,33)	NC	2362	0.3	2.6	0.07	0.15	0.22	2.0	9.2	0.15	0.23	0.38
	WC	1864	0.4	5.4	0.06	0.08	0.10	3.9	14.3	0.13	0.21	0.38
	SC	1266	0.2	2.6	0.06	0.09	0.12	2.8	19.1	0.12	0.17	0.31
(100,278,66)	NC	11 795	1.3	11.1	0.08	0.12	0.19	8.9	42.1	0.11	0.20	0.27
	WC	9581	1.7	20.5	0.06	0.09	0.10	16.5	82.1	0.16	0.19	0.22
	SC	5265	2.8	58.4	0.02	0.06	0.08	45.3	227.5	0.12	0.20	0.32
(150,429,100)	NC	32 346	6.5	78.8	0.06	0.10	0.13	121.6	314.1	0.20	0.33	0.46
	WC	25 561	6.0	90.0	0.04	0.07	0.09	116.5	318.9	0.28	0.34	0.50
	SC	13 193	8.7	141.4	0.04	0.06	0.07	224.3	830.0	0.15	0.31	0.45
(200,573.6,133)	NC	68 979	24.3	247.5	0.06	0.10	0.14	288.8	941.1	0.13	0.30	0.47
	WC	52 742	22.5	292.4	0.05	0.08	0.12	367.7	1147.0	0.23	0.33	0.42
	SC	23 798	28.4	412.0	0.04	0.08	0.11	963.4	2305.5	0.31	0.40	0.53
(200,582,133) ^a	NC	50 368	327.4	1228.7			0.27	5071.6	3112.2		1.63	
	WC	38 748	76.5	787.8			0.10	975.6	1443.6		0.68	
	SC	24 758	37.2	469.3			0.08	720.4	1117.8		0.64	
(250,714,166)	NC	116 300	51.3	482.0	0.09	0.11	0.13	686.8	1371.7	0.23	0.40	0.68
	WC	92 500	43.2	463.9	0.07	0.11	0.18	775.7	1428.0	0.30	0.50	0.70
	SC	37 254	29.9	420.5	0.05	0.07	0.11	1218.7	3531.1	0.27	0.35	0.47
(100,278,33)	NC	2994	0.2	3.2	0.05	0.07	0.10	0.4	5.9	0.06	0.08	0.10
	WC	2400	0.2	3.9	0.05	0.06	0.10	0.5	10.4	0.03	0.07	0.14
	SC	1315	0.3	9.7	0.02	0.03	0.06	1.1	16.1	0.04	0.06	0.10
(200,574,66)	NC	16 954	1.6	55.6	0.02	0.03	0.04	8.2	112.7	0.03	0.07	0.09
	WC	13 076	1.5	45.9	0.03	0.03	0.04	11.8	156.0	0.05	0.08	0.12
	SC	5806	1.4	58.7	0.01	0.02	0.04	15.3	211.7	0.05	0.07	0.10
(300,869,100)	NC	50 418	6.5	175.9	0.03	0.04	0.06	31.8	362.0	0.05	0.09	0.12
	WC	39 084	4.8	166.1	0.02	0.03	0.04	28.9	466.1	0.03	0.09	0.14
	SC	14 545	2.8	163.3	0.01	0.02	0.02	49.2	767.5	0.04	0.07	0.10
(400,1165,133)	NC	103 356	19.5	458.4	0.03	0.05	0.06	179.2	1331.8	0.10	0.15	0.26
	WC	77 037	22.3	540.2	0.03	0.05	0.06	208.2	1833.5	0.08	0.13	0.18
	SC	26 583	10.7	315.1	0.03	0.04	0.04	215.7	1909.7	0.07	0.13	0.19
(500,1458,166)	NC	196 267	49.9	1093.8	0.04	0.05	0.08	373.5	2801.9	0.12	0.14	0.17
	WC	145 782	61.8	1697.3	0.03	0.04	0.04	821.8	6150.6	0.10	0.13	0.15
	SC	44 751	22.9	1392.3	0.01	0.02	0.04	587.0	6190.8	0.10	0.11	0.14

Abbreviation: VRPTW = vehicle routing problem with time windows.
LET and NEWLET instances.

Allowing nonelementary routes in the pricing problem could be advantageous for the road-network approach in terms of computing time, however, it has a negative impact on the quality of the lower bound and thereafter could lead to longest branching times. Increasing the density of customers in the road network is also advantageous for the road-network approach. Another factor is the width of time windows. We observe that when time windows are wider, the growth in the number of feasible label extensions is more important with the road-network. Thus, the acceleration factor of the multigraph-based column generation compared to the road-network-graph-based column generation is larger. We also show that introducing additional resources, like vehicle capacity, influences the relative efficiency of the two approaches. Experiments on the m-TSPTW and on the VRPTW show that the increase in computing times is stronger with the road-network graph.

Experimentations on real road-networks (AIX instances) are even more in favor of the multigraph. When the road-network is large (thousands of nodes and arcs), searching for new columns in the road-network graph becomes very detrimental. Note that, computing the multigraph efficiently remains possible, as shown in [3].

TABLE 6 LP relaxation with elementary routes for the VRPTW (NEWLET instances with $|V| = 250$)

C	Corr	$ A^{MG} $	Narrow time windows						Wide time windows				
			CPU (s)			$\frac{CPU_{MG}}{CPU_{RN}}$			CPU (s)		$\frac{CPU_{MG}}{CPU_{RN}}$		
			MG	RN	Min	Avg	Max	MG	RN	Min	Avg	Max	
25	NC	2553	0.1	7.8	0.01	0.02	0.03	0.2	13.8	0.01	0.02	0.03	
	WC	2123	0.2	12.1	0.01	0.02	0.02	0.3	15.1	0.02	0.02	0.03	
	SC	878	0.1	9.3	0.01	0.01	0.02	0.2	18.9	0.01	0.01	0.03	
50	NC	10 704	0.6	32.5	0.01	0.02	0.04	2.3	68.9	0.02	0.03	0.05	
	WC	8697	0.7	31.8	0.02	0.02	0.03	3.5	129.5	0.02	0.03	0.04	
	SC	3478	0.4	30.9	0.01	0.01	0.02	3.4	97.4	0.02	0.04	0.06	
75	NC	24 267	2.4	78.2	0.02	0.03	0.04	8.8	164.7	0.04	0.06	0.09	
	WC	19 333	2.5	67.7	0.02	0.04	0.05	19.1	287.6	0.06	0.08	0.10	
	SC	7815	1.6	76.2	0.01	0.02	0.03	22.3	320.4	0.04	0.07	0.10	
100	NC	42 367	7.0	126.3	0.04	0.06	0.08	29.4	313.5	0.08	0.10	0.14	
	WC	34 163	6.4	133.1	0.03	0.05	0.06	86.2	588.6	0.11	0.14	0.16	
	SC	13 716	4.8	128.0	0.03	0.04	0.05	130.3	816.7	0.07	0.15	0.22	
125	NC	66 248	19.8	245.0	0.05	0.08	0.10	92.8	538.4	0.12	0.17	0.23	
	WC	52 534	14.2	199.0	0.06	0.07	0.08	179.6	759.9	0.19	0.25	0.30	
	SC	21 336	9.3	180.4	0.04	0.05	0.06	255.4	1495.9	0.09	0.18	0.26	
166	NC	116 300	51.3	482.0	0.09	0.11	0.13	686.8	1371.7	0.23	0.40	0.68	
	WC	92 500	43.2	463.9	0.07	0.11	0.18	775.7	1428.0	0.30	0.50	0.70	
	SC	37 254	29.9	420.5	0.05	0.07	0.11	1218.7	3531.1	0.27	0.35	0.47	

Abbreviation: VRPTW = vehicle routing problem with time windows.

The obtained results thus show that for a large panel of test instances, it is consistently more interesting to tackle the problem using the multigraph, which is in contrast with the conclusions drawn by Letchford et al. [15]. Many reasons can explain this discrepancy.

First, Letchford et al. [15] generate only one route with negative reduced cost at each iteration of the column generation, while in our experiments (following common practices) all nondominated routes with negative reduced cost are generated for each iteration. Secondly, computational experiments in [15] are based on a single instance for each parameter configuration, which limits the scope of their conclusions. In addition, Letchford et al. [15] only consider instances with high densities of customers, which is not representative for real road-networks. Finally, by comparing our results to those reported in [15], we observe that the computing times are extremely high in Letchford et al. [15] for the multigraph approach. It might be due to the nature of data structures and details of implementations that are not adapted to the multigraph setting.

5 | CONCLUSIONS

In this paper, we investigated routing problems with road-network information for the VRPTW_{RN}. We identified two possible models to capture this information: multigraph or road-network graph.

In this paper, we elaborated on the road network approach and complemented the results presented in [15]. We proposed a complete branch-and-price algorithm based on the road-network graph. We conducted computational experiments based on different types of instances to compare this algorithm to a similar solution scheme relying on the multigraph. We analyzed the impact of different instance characteristics, such as customer density in the road network, time and capacity constraints, on the relative efficiency of the two algorithms. Results showed a clear advantage for the multigraph, especially for the more realistic instances. Results however show that using the road-network is also a tractable option.

In future works, we plan to investigate other types of VRP for which the customer-based graph shows some limits. Many examples are given in Ben et al. [4], including problems with multiple attributes, but also problems with complex cost functions or constraints (e.g., energy consumption for electric vehicles, time-dependent or stochastic travel times, management of driver's working-hour regulation). In some problems, the only possible option could be the road-network graph, which motivates the development of more efficient solution schemes for this case.

TABLE 7 Branch-and-price with elementary routes for the VRPTW

(V , A , C)	Corr	A ^{MG}	Narrow time windows						Wide time windows							
			Solved		CPU (s)		CPU _{MG} CPU _{RN}		Solved		CPU (s)		CPU _{MG} CPU _{RN}			
			MG	RN	MG	RN	Min	Avg	Max	MG	RN	MG	RN	Min	Avg	Max
(50,135,33)	NC	2362	5	5	0.4	3.0	0.07	0.15	0.22	5	5	11.1	41.5	0.21	0.33	0.46
	WC	1864	5	5	0.8	12.3	0.05	0.08	0.13	5	4	132.0	386.0	0.03	0.25	0.49
	SC	1266	5	5	0.3	2.4	0.08	0.11	0.14	5	4	151.5	509.4	0.14	0.19	0.25
(100,278,66)	NC	11 795	5	5	1.4	10.9	0.10	0.13	0.18	5	4	110.0	153.1	0.02	0.19	0.29
	WC	9581	5	5	3.3	141.2	0.01	0.10	0.16	4	3	216.9	529.6	0.14	0.16	0.20
	SC	5265	5	4	786.2	1311.2	0.00	0.04	0.07	0	0	7200.0	7200.0	–	–	–
(150,429,100)	NC	32 346	5	5	7.3	77.4	0.06	0.11	0.15	1	1	314.0	2285.0	0.14	0.14	0.14
	WC	25 561	5	5	12.1	112.9	0.04	0.09	0.15	1	0	4289.8	7200.0	–	–	–
	SC	13 193	5	5	17.5	265.8	0.04	0.07	0.08	0	0	7200.0	7200.0	–	–	–
(200,573.6,133)	NC	68 979	5	5	32.7	296.1	0.06	0.12	0.20	2	1	930.8	940.7	0.12	0.12	0.12
	WC	52 742	5	4	123.0	1096.4	0.03	0.07	0.10	0	0	7200.0	7200.0	–	–	–
	SC	23 798	5	4	208.9	754.1	0.06	0.10	0.15	0	0	7200.0	7200.0	–	–	–
(200,582,133) ^a	NC	50 368	1	1	767.5	4208.2	–	0.18	–	0	0	7200	7200	–	–	–
	WC	38 748	1	0	143.4	7200.0	–	–	–	0	0	7200	7200	–	–	–
	SC	24 758	0	0	7200.0	7200.0	–	–	–	0	0	7200	7200	–	–	–
(250,714,166)	NC	116 300	5	3	183.1	1821.8	0.04	0.10	0.14	1	1	159.1	2258.3	0.07	0.07	0.07
	WC	92 500	5	5	129.9	682.8	0.07	0.15	0.32	0	0	7200.0	7200.0	–	–	–
	SC	37 254	5	3	1125.8	2270.5	0.06	0.08	0.09	0	0	7200.0	7200.0	–	–	–
(100,278,33)	NC	2994	5	5	0.2	2.5	0.08	0.10	0.12	5	5	1.4	162.3	0.00	0.08	0.12
	WC	2400	5	5	0.3	4.0	0.05	0.08	0.12	5	5	1.0	235.2	0.00	0.05	0.11
	SC	1315	5	5	0.3	7.9	0.03	0.05	0.08	5	5	2.7	142.8	0.02	0.06	0.10
(200,574,66)	NC	16 954	5	5	2.2	58.2	0.02	0.03	0.05	5	4	40.9	352.4	0.02	0.07	0.11
	WC	13 076	5	5	1.7	1106.5	0.00	0.03	0.04	3	3	53.8	1016.5	0.04	0.05	0.06
	SC	5806	5	5	2.1	63.6	0.01	0.03	0.07	4	2	149.4	6155.7	0.01	0.01	0.02
(300,869,100)	NC	50 418	5	5	10.7	184.3	0.05	0.06	0.10	5	4	137.4	499.0	0.07	0.13	0.18
	WC	39 084	5	4	8.7	213.0	0.04	0.05	0.05	3	1	256.8	1379.7	0.19	0.19	0.19
	SC	14 545	5	5	5.9	405.5	0.01	0.02	0.03	3	1	1466.8	672.3	0.06	0.06	0.06
(400,1165,133)	NC	103 356	5	4	66.4	1981.6	0.01	0.06	0.10	4	3	247.7	1499.4	0.11	0.15	0.20
	WC	77 037	5	5	89.6	1338.5	0.04	0.07	0.12	3	0	4572.8	7200.0	–	–	–
	SC	26 583	5	3	39.0	848.5	0.04	0.04	0.05	0	0	7200.0	7200.0	–	–	–
(500,1458,166)	NC	196 267	5	3	102.8	1775.7	0.04	0.07	0.10	3	1	3005.8	2934.6	0.19	0.19	0.19
	WC	145 782	5	4	241.4	1664.2	0.04	0.05	0.06	1	0	1643.0	7200.0	–	–	–
	SC	44 751	5	3	56.5	4596.4	0.00	0.02	0.03	0	0	7200.0	7200.0	–	–	–

Abbreviation: VRPTW = vehicle routing problem with time windows.
 No solution have been found within the time limit.
 LET and NEWLET instances.

Another interesting perspective would be to evaluate the adaptability to the road-network context of the large number of tools developed for the solution of VRP with branch-and-price (and-cut): subset-row inequalities, ng-paths, and so on. Also, new techniques specifically designed for the road-graph case might be developed. Maybe, with more advanced implementation of the branch-and-price and new techniques, opposite results could be observed, and the road-network could appear as a better choice than the multigraph.

A final perspective is to develop approaches that would mix, in some fashion that has to be defined, the three types of graphs: customer-based graph, multigraph and road-network graph.

We believe that pursuing this stream of research is especially important regarding the new challenges faced by vehicle routing optimization. The pressure for efficient and carbon-aware city distribution systems, the availability of huge amount of data, the possibility offered by this data to develop stochastic/dynamic models, makes the traditional customer-based graph increasingly questionable. Developing efficient solution methods for the two alternative models, road-network graph and multigraph, and identifying the respective pros and cons of these models could have a deep impact on the development of modern distribution systems.

TABLE 8 Branch-and-price with elementary routes for the VRPTW (NEWLET instances with $|V|=250$)

C	Corr	A ^{MG}	Narrow time windows							Wide time windows						
			Solved		CPU (s)		$\frac{CPU_{MG}}{CPU_{RN}}$			Solved		CPU (s)		$\frac{CPU_{MG}}{CPU_{RN}}$		
			MG	RN	MG	RN	Min	Avg	Max	MG	RN	MG	RN	Min	Avg	Max
25	NC	2553	5	5	0.2	9.7	0.01	0.02	0.03	5	5	0.2	14.9	0.01	0.02	0.02
	WC	2123	5	5	0.2	12.4	0.01	0.02	0.02	5	4	0.4	20.1	0.01	0.02	0.03
	SC	878	5	5	0.1	11.7	0.01	0.01	0.02	5	5	0.3	21.9	0.01	0.02	0.04
50	NC	10704	5	5	0.7	35.2	0.01	0.02	0.03	5	5	9.7	1167.8	0.01	0.03	0.06
	WC	8697	5	5	0.8	41.6	0.01	0.02	0.03	5	4	37.3	1219.3	0.02	0.03	0.03
	SC	3478	5	5	0.8	296.6	0.00	0.01	0.02	5	3	843.4	1080.7	0.02	0.04	0.04
75	NC	24267	5	5	2.6	78.4	0.03	0.04	0.06	4	4	9.7	179.7	0.03	0.05	0.08
	WC	19333	5	5	2.4	71.9	0.03	0.04	0.04	3	3	13.0	151.3	0.08	0.08	0.10
	SC	7815	5	4	3.8	82.8	0.01	0.02	0.02	1	1	28.1	665.7	0.04	0.04	0.04
100	NC	42367	5	5	12.5	312.4	0.03	0.06	0.11	5	4	106.8	562.8	0.06	0.16	0.25
	WC	34163	5	5	7.1	144.4	0.04	0.05	0.06	2	2	1505.9	2091.2	0.13	0.46	0.80
	SC	13716	5	4	274.7	197.9	0.02	0.85	3.33	0	0	7200.0	7200.0	-	-	-
125	NC	66248	5	5	35.0	439.1	0.06	0.09	0.16	3	1	1285.8	241.1	0.14	0.14	0.14
	WC	52534	5	5	17.2	289.8	0.04	0.06	0.09	1	1	101.1	399.4	0.25	0.25	0.25
	SC	21336	5	4	10.8	217.3	0.03	0.04	0.05	0	0	7200.0	7200.0	-	-	-
166	NC	116300	5	3	183.1	1821.8	0.04	0.10	0.14	1	1	159.1	2258.3	0.07	0.07	0.07
	WC	92500	5	5	129.9	682.8	0.07	0.15	0.32	0	0	7200.0	7200.0	-	-	-
	SC	37254	5	3	1125.8	2270.5	0.06	0.08	0.09	0	0	7200.0	7200.0	-	-	-

Abbreviation: VRPTW = vehicle routing problem with time windows.
No solution have been found within the time limit.

TABLE 9 Branch-and-price with elementary routes for the VRPTW (AIX instances: $|V|=5437$, $|A|=10181$)

C	Inst.	CPU_{MG}	CPU_{RN}	$\frac{CPU_{MG}}{CPU_{RN}}$	Sav_{MC}	Sav_{MT}
5	1	0.05	5.48	0.009	17.6	7.3
	2	0.07	4.16	0.016	12.0	4.7
	3	0.05	4.28	0.011	0.0	17.4
	4	0.06	8.77	0.007	0.0	11.8
	5	0.05	14.84	0.003	0.0	9.4
10	1	0.09	11.87	0.008	8.7	7.0
	2	0.08	7.08	0.011	6.2	5.8
	3	0.07	6.03	0.011	0.0	11.9
	4	0.09	11.57	0.007	0.5	6.8
	5	0.08	25.65	0.003	0.0	11.5
25	1	0.20	56.85	0.004	3.8	7.3
	2	0.20	51.35	0.004	1.9	6.0
	3	0.17	35.10	0.005	6.5	7.9
	4	0.39	111.84	0.004	5.1	13.0
	5	0.18	80.95	0.002	11.2	9.5
50	1	0.99	113.46	0.009	5.6	16.2
	2	3.33	7200.00	-	1.1	7.4
	3	2.11	147.63	0.014	1.1	6.3
	4	1.03	252.03	0.004	8.5	10.8
	5	17.36	7200.00	-	5.5	5.5

Abbreviation: VRPTW = vehicle routing problem with time windows.

ACKNOWLEDGMENTS

The first author was supported by the Labex IMobS3, by the European Fund for Regional Development (FEDER Auvergne region) and by the Auvergne Region.

ORCID

Hamza Ben Ticha  <https://orcid.org/0000-0003-4328-777X>

Dominique Feillet  <http://orcid.org/0000-0003-1246-223X>

REFERENCES

- [1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W. Savelsbergh, and P.H. Vance, *Branch-and-price: Column generation for solving huge integer programs*, *Oper. Res.* **46**(3) (1998), 316–329.
- [2] H. Ben Ticha, N. Absi, D. Feillet, and A. Quilliot, *Adaptive Large Neighborhood Search for the Vehicle Routing Problem with Time Windows with a Multigraph Representation for the Road Network*, Tech. Rep. EMSE CMP-SFL 2017/7, Ecole des Mines de Saint Etienne, CMP, Gardanne, France (2017a).
- [3] H. Ben Ticha, N. Absi, D. Feillet, and A. Quilliot, *A Solution Method for the Multidestination Bi-objectives Shortest Path Problem*, Tech. Rep. EMSE CMP-SFL 2017/5, Ecole des Mines de Saint Etienne, CMP, Gardanne, France (2017b).
- [4] H. Ben Ticha, N. Absi, D. Feillet, and A. Quilliot, *Vehicle routing problems with road-network information: State of the art*, *Networks* (2018), 1–14. <https://doi.org/10.1002/net.21808>.
- [5] C. Bode and S. Irnich, *Cut-first branch-and-price-second for the capacitated arc-routing problem*, *Oper. Res.* **60**(5) (2012), 1167–1182.
- [6] J.-F. Cordeau, M. Gendreau, and G. Laporte, *A tabu search heuristic for periodic and multi-depot vehicle routing problems*, *Networks* **30**(2) (1997), 105–119.
- [7] G.B. Dantzig and J.H. Ramser, *The truck dispatching problem*, *Manag. Sci.* **6**(1) (1959), 80–91.
- [8] M. Desrochers, J. Desrosiers, and M. Solomon, *A new optimization algorithm for the vehicle routing problem with time windows*, *Oper. Res.* **40**(2) (1992), 342–354.
- [9] D. Feillet, *A tutorial on column generation and branch-and-price for vehicle routing problems*, *4OR* **8**(4) (2010), 407–424.
- [10] 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* **44**(3) (2004), 216–229.
- [11] T. Garaix, C. Artigues, D. Feillet, and D. Josselin, *Vehicle routing problems with alternative paths: An application to on-demand transportation*, *Eur. J. Oper. Res.* **204**(1) (2010), 62–75.
- [12] B.L. Golden, S. Raghavan, and E.A. Wasil, *The vehicle routing problem: Latest advances and new challenges*, vol. **43**, Springer Science & Business Media (2008).
- [13] S. Irnich and G. Desaulniers, “*Shortest path problems with resource constraints*,” *Column Generation*, G. Desaulniers, J. Desrosiers, M.M. Solomon (eds.), Springer, Boston (2005), pp. 33–65.
- [14] G. Laporte, *Fifty years of vehicle routing*, *Transport. Sci.* **43**(4) (2009), 408–416.
- [15] A.N. Letchford, S.D. Nasiri, and A. Oukil, *Pricing routines for vehicle routing with time windows on road networks*, *Comput. Oper. Res.* **51** (2014), 331–337.
- [16] A.N. Letchford and A. Oukil, *Exploiting sparsity in pricing routines for the capacitated arc routing problem*, *Comput. Oper. Res.* **36**(7) (2009), 2320–2327.
- [17] G. Righini and M. Salani, *New dynamic programming algorithms for the resource constrained elementary shortest path problem*, *Networks* **51**(3) (2008), 155–170.
- [18] P. Serafini, “*Some considerations about computational complexity for multi objective combinatorial problems*,” *Recent Advances and Historical Development of Vector Optimization*, Springer (1987), pp. 222–232.
- [19] M.M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, *Oper. Res.* **35**(2) (1987), 254–265.
- [20] H.B. Ticha, N. Absi, D. Feillet, and A. Quilliot, *Empirical analysis for the VRPTW with a multigraph representation for the road network*, *Comput. Oper. Res.* **88** (2017), 103–116.
- [21] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*, SIAM, Philadelphia, 2014.

How to cite this article: Ben Ticha H, Absi N, Feillet D, Quilliot A, Van Woensel T. A branch-and-price algorithm for the vehicle routing problem with time windows on a road network. *Networks*. 2019;73:401–417. <https://doi.org/10.1002/net.21852>