The Time-Dependent Vehicle Routing Problem with Time Windows and Road-Network Information

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

ORIGINAL RESEARCH

The Time-Dependent Vehicle Routing Problem
with Time Windows and Road-Network Information

Hamza Ben Ticha[1] · Nabil Absi[1] · Dominique Feillet[1] (ORCID) **· Alain Quilliot[2] · Tom Van Woensel[3]**

Abstract

Most time-dependent vehicle routing problems are based on a similar modeling paradigm: travel time information is represented by travel time functions between pairs of points of interest (e.g., depot or customers). Only a few papers investigate how these functions can be computed using the available travel time information. Furthermore, most of them neglect the possibility that different paths could be selected in the road network depending on the compromises they offer between cost (distance) and travel time. In this paper, we propose a new setting where travel time functions are defined on road-network arcs. We consider the Time-Dependent Vehicle Routing Problem with Time Windows and solve it with a branch-and-price algorithm. As far as we know, this is the first exact approach for a time-dependent vehicle routing problem when travel time functions are initially defined on the segments of a road-network.

Keywords Time-Dependent Vehicle Routing Problem · Branch-and-price · Road network

1 Introduction

Vehicle routing problems (VRPs) define a class of combinatorial optimization problems that aim at computing a minimum-cost set of routes for a fleet of capacitated vehicles to serve a set of customers. In general, every customer is served exactly once and every route starts and ends at a central depot. The total demand delivered along a route should not exceed the vehicle capacity. Due to their numerous applications,

---

VRPs are widely studied in the operations research literature (see, e.g., Laporte [28], Golden et al. [20], Toth and Vigo [36]).

In most models, the travel time between two customer locations is invariant over time. Clearly, in real-life, this assumption does not hold, in particular in urban areas, where travel times and speeds are subject to significant variations over the day. These variations may be due to predictable events such as traffic congestion and unpredictable events such as accidents, weather conditions, and vehicle breakdowns. In the literature, predictable variations are addressed using models that incorporate time-dependent travel information (see, e.g., Ichoua et al. [23], Fleischmann et al. [14], Kok et al. [26], Donati et al. [9], Figliozzi [13], Dabia et al. [7]).

Typically, VRPs, including those with or without time-dependent travel times, are tackled using a so-called customer-based graph. In a customer-based graph, each node represents a point of interest, generally a customer or the depot. Arcs indicate the order in which nodes are visited. Implicitly, an arc is also an abstraction of a precomputed path in the underlying road network. The customer-based-graph model relies on the assumption that the best paths can easily be precomputed. In many situations, this assumption is not valid (Ben Ticha et al. [4]). One of these situations is when several attributes (like travel time and distance) are defined on road segments. In this case, alternative paths with different compromises may exist between pairs of points in the road network. Discarding these alternatives when tackling VRPs can increase solution costs significantly (Garaix et al. [15], Ben Ticha et al. [3]).

This situation is depicted in Fig. 1. The figure shows a road network, represented with a road-network graph, and two customer-based graphs. Arcs in the min-cost (resp., min-time) graph represent shortest paths in distance (resp., time). Route (0,1,2,3,0), for example, has a cost 14 in the min-cost graph and 18 in the min-time graph. On the contrary, if a time window at customer 1 imposes the vehicle to arrive before time 4, the route is not feasible in the min-cost graph, while it is still feasible in the min-time graph. Hence, one cannot know in advance which customer-based graph is more appropriate.

In the literature, few papers developed innovative methodologies to counter the negative effects of customer-based graphs. Two models are proposed. The first model consists in representing the road network with a multigraph. In this model, all efficient paths between two points of interest are considered and maintained when solving the problem [2, 3, 15, 27]. The second approach consists in tackling the problem directly on a graph that mimics the original road network, called the road-network graph (Letchford et al. [29]).

Figure 1 represents a road-network graph and the corresponding multigraph. In this graph, the best feasible route is always present. For example, route (0, 1, 2, 3, 0) can be executed by reaching customer 1 before time 4 and still having a total cost of 15 instead of 18 in the min-time graph. If there is no time constraint, a total cost of 14 can be achieved, like in the min-cost graph.

In this paper, we investigate alternatives to the customer-based graph for time-dependent VRPs. Intuitively, this is important since the road-network graph better captures congestion effects and gives more contrasted characteristics between efficient paths. The most relevant study with that matter is proposed by Huang et al. [22]. The authors are interested in the solution of a time-dependent VRP and adopt

a multigraph modeling: a set of efficient paths from the road network is considered between each pair of points of interest. A limitation of this study is however the computation of the multigraph. As detailed in the literature section, only a small subset of efficient paths is considered in that paper.

Computing exactly the multigraph with time-dependent travel time information is extremely difficult or even intractable. Indeed, one has to compute all efficient paths in the road network, for each pair of points of interest and for each possible departure time, which induces the solution of many NP-hard multicriteria shortest path problems (see Serafini [33], Hansen [21]). For this reason, we focus in this study on the solution of a time-dependent VRP expressed on a road-network graph. We select the Time-Dependent Vehicle Routing Problem with Time Windows (TDVRPTW) as a test-bed problem and investigate the methodology that can be developed to solve exactly the problem. We then numerically evaluate how efficient it is, and what are the gains compared to the standard modeling with a customer-based graph.

We coin the problem as TDVRPTW$_{RN}$ because detailed road-network (RN) information is used. In this problem, a road-network graph is given, with travel speed functions assigned to every arc. Time windows restrict visit times for customers. The objective is to minimize the traveled distance. We develop a branch-and-price algorithm able to solve exactly the TDVRPTW$_{RN}$. We compare our solutions with those found on two customer-based graphs obtained from the road-network graph: a min-cost graph where paths are selected according to their travel distance, a min-time graph where paths are selected according to their travel time (see Fig. 1). Note that in both cases we do not change the objective, that is still to minimize the traveled distance. We base our experiments on benchmark instances simulating small real-life road networks, following a procedure proposed by Letchford et al. [29].

It is important to highlight that details on how travel time functions can be obtained for customer-based graphs are missing in most papers on time-dependent VRPs. Conventionally, in these papers, the customer-based graph is introduced first. Then, time-varying speed profiles are introduced on every arc (of the customer-based graph). These speed profiles are finally converted into travel times (see, e.g., Ichoua et al. [23], Jabali et al. [24], Cordeau et al. [6]). With this approach, speeds or travel times are not explicitly defined on road network arcs. In practice, it should be the opposite. Travel time/speed information is obtained from geographic information systems, where it is defined on road-network arcs. Travel time functions for the customer-based graphs should thus be computed using this source of information. Only a few papers are interested in how it can be done (see, e.g., Ghiani and Guerriero [17] or the recent PhD thesis by Gmira [18]). Several of these studies are not concerned with vehicle routing. They aim at computing point-to-point fastest paths for a given starting time (see, e.g., Delling and Wagner [8], Orda and Rom [30], Kaufman and Smith [25]).

This paper makes several contributions to the literature:

The main contribution is to initiate the solution process from travel speed inputs defined on the road network. Then, it becomes clear that the problem cannot be solved exactly using a customer-based graph. To handle this issue, we propose to address the problem directly on the road-network graph and we develop a branch-and-price scheme for the TDVRPTW$_{RN}$. As far as we know, this is the first approach
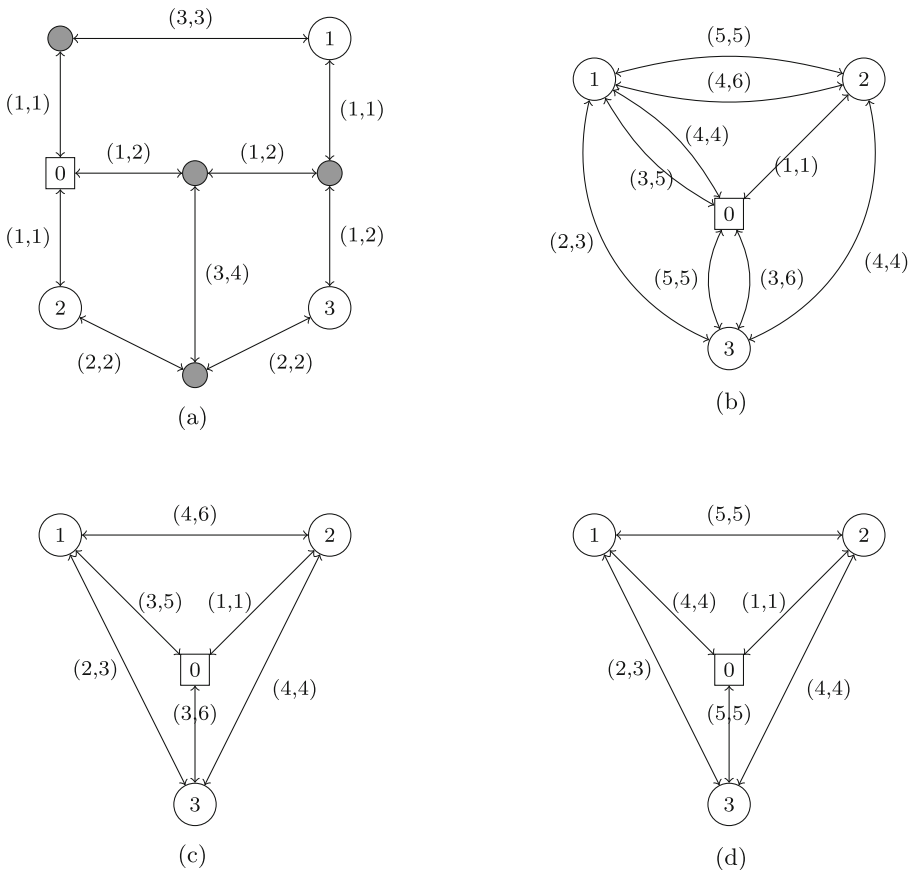
Fig. 1 Graphs for a simple example with 3 customers (arcs labeled with (distance,time))

that guarantees to find optimal solutions in this context. We are only aware of the work by Gmira et al. [19] following the same line of research. They address the same problem, also named TDVRPTW$_{RN}$, with the sole difference that the objective is to minimize travel time. Instead of investigating the exact solution of the problem, they propose a tabu search algorithm.

The second contribution of this paper is to numerically evaluate how considering road-network information complicates the resolution and how it impacts solution qualities. Our experiments complete and confirm results obtained for the VRPTW in Ben Ticha et al. [5]. They suggest that using the road-network can be computationally penalizing for large-scale road networks and remains manageable otherwise. On the other hand, it allows important cost savings against searching a set of optimal routes in the min-time graph (which is the only customer-based graph that always ensures finding a solution if one exists). Gaps are more limited if the min-cost graph is used: searching an optimal set of routes in this graph is apparently a good heuristic. However, experiments also show that using the min-cost graph relatively often leads to

false infeasible (i.e., instances for which no solution exists using the min-cost graph while the problem is feasible).

A final contribution is to help to clarify how customer-based graphs can be computed. This information can never be found in the time-dependent VRP literature. Note in particular that in this literature, travel times are time-dependent but distances usually are not. This is not true in a min-time graph where the fastest road-network path between two points of interest can vary during the day. We report two dedicated algorithms, for the min-cost and min-time graphs, respectively.

In the remainder of this paper, we first review the relevant literature in Section 2. We then formally describe the TDVRPTW$_{RN}$ (Section 3). In Section 4, we focus on the construction of customer-based graphs. The branch-and-price algorithm developed for the solution of the TDVRPTW$_{RN}$ is presented in Section 5. Finally, we present computational experiments and comment results in Section 6.

2 Literature Review

Despite the huge number of papers dealing with VRPs, the number of papers addressing time-dependent problems is relatively limited. As far as we know, the first study dealing with a vehicle routing problem where travel times vary over time is by Beasley [1]. Interested readers can refer to Gendreau et al. [16] for a recent survey and for a discussion on the different models and problems that arise from time-dependent data.

As already stated, papers on time-dependent vehicle routing generally assume the data given. A stream of papers however examines the issue of constructing a database for travel times on time-dependent road networks. Eglese et al. [10], for example, propose a model that constructs a Road Timetable based on historical data and provides the shortest travel time for different departure times. They illustrate the benefits of considering the Road Timetable information when tackling a VRP. The main drawback of their approach is that fastest paths are computed only for a set of specified departure times. Then, the travel time for any departure time is approximated using these computed values. This procedure may thus provide a weak estimation of travel times, with consequences on solution quality or feasibility.

Time-dependent VRPs are most of the time tackled on customer-based graphs. In the literature, an increasing number of papers investigates the negative impact that this graph can have on solution quality for non-time-dependent vehicle routing problems. Garaix et al. [15] were the first to point out that transforming road-network information into a customer-based graph can result in losing solution optimality. They propose to replace the customer-based graph with a multigraph in which all efficient paths are maintained. This modeling approach is investigated more in-depth by Ben Ticha et al. [3]. They propose a branch-and-price algorithm and present an extensive computational study based on instances from the literature and instances representing real road networks. Reported results confirm the negative impact of the customer-based graph. Letchford et al. [29] suggest that applying the branch-and-price framework to a road-network graph would be more efficient. However, they limit their investigations to the pricing problem. More recently, Ben Ticha et al. [5]

revisited the results presented by Letchford et al. [29]. They propose a complete branch-and-price scheme and conduct extensive comparisons between the multigraph and the road-network graph approaches. Their results contradict those presented in Letchford et al. [29] and conclude that in most cases the multigraph-based branch-and-price algorithm is more efficient.

The aforementioned papers confirm that when several attributes are defined on road segments, the traditional customer-graph affects solution quality for VRPs. In the time-dependent VRPs literature, little attention has been paid to this issue. Apart from the recent paper by Gmira et al. [19], the most relevant study is proposed by Huang et al. [22]. They introduce the so-called Time-Dependent Vehicle Routing Problem with Path Flexibility (TDVRP-PF). Path Flexibility means that when solving the problem a set of alternative paths between each pair of nodes is maintained in a multigraph structure. These alternative paths present different compromises in terms of travel time and distance and are computed as follows: a modified Dijkstra's algorithm is first applied to compute the fastest paths for a set of discretized departure times then the shortest path is included into the set of considered paths. Huang et al. [22] propose a mathematical formulation for the TDVRP-PF where the selection of a path between two customer nodes is embedded as a decision variable. They present a computational study based on instances derived from the road network of Beijing, China. Numerical results show that the path flexibility improves significantly the solution quality in terms of cost (up to 5%) and fuel consumption (up to 7%). The authors observe that these improvements are more significant than those obtained using flexible departure times at the depot and customer nodes. They also investigate the TDVRP-PF under stochastic traffic conditions and note that path flexibility provides a natural recourse in this case.

Setak et al. [31] also introduce a multigraph to tackle a problem that they call Time-Dependent Pollution Routing Problem. In their model, parallel arcs present different compromises on travel times, energy consumption, and toll costs. Unfortunately, they do not specify how these arcs are computed. To solve the problem, they propose a tabu search heuristic. Their computational experiments show that using the multigraph, an average gain of 1.1% is achieved on solution costs compared to costs obtained with a customer-based graph. Later, Tikani and Setak [35] presented multigraph networks in the application area of emergency logistics management.

Besides the papers cited above, similar modelings are used but with relatively different goals. Setak et al. [34] investigate a time-dependent vehicle routing problem defined on a multigraph where parallel arcs represent different travel speed distributions and satisfy the FIFO property. Wang and Lee [37] introduce the so-called Time-Dependent Alternative Vehicle Routing Problem, where pairs of nodes are connected with two arcs. The first arc is defined with a time-dependent travel speed distribution and is used when traffic is low. The second arc is defined with constant travel time and could be used as an alternative during peak hours. In both studies, parallel arcs do not provide different compromises but, instead, offer different alternatives to be used depending on the departure time: at any point in time, one arc dominates the other.

As far as we know, Gmira et al. [19] are the only ones that fully consider all the alternatives offered by the road network for a time-dependent VRP. Contrary to the

previous authors, they do not introduce a multigraph. They propose a tabu search algorithm, with a specialized CROSS operator. When sequences of customers are swapped, feasibility is checked with a special structure named d̂ominant shortest path structure and solution value is only approximated. With appropriate incremental data management, it allows evaluating neighborhood solutions in constant time.

3 Problem Description

The Time-Dependent Vehicle Routing Problem With Time Windows and Road-Network Information (TDVRPTW$_{RN}$) is defined on a directed road-network graph $G^{RN} = (V^{RN}, A^{RN})$. V^{RN} contains the depot node 0 and nodes that represent road junctions. Among these nodes, a subset C represents customers. An arc $(i, j) \in A^{RN}$ models a road segment and is defined with a travel distance (road segment length) d_{ij} and a travel time function $\tau_{ij}(t)$ which indicates the time needed to go through arc (i, j) for a given departure time at node i. To every customer $i \in C$ are assigned a time window $[e_i, l_i]$, a service time st_i and a demand q_i. The time horizon $[0, H]$ is given by the depot time window $[e_0, l_0]$. A fleet of K homogeneous vehicles with a finite capacity Q is available to perform the deliveries.

The TDVRPTW$_{RN}$ consists in designing a set of routes that minimizes the total traveled distance for the fleet of vehicles. The total load delivered along a route cannot exceed the vehicle capacity. Each customer has to be served exactly once and within its time window. Customer nodes can however be traversed several times, like any other road junctions, without service.

Travel time functions τ_{ij} are obtained from speed profiles. Speed profiles are defined by step-wise functions that divide the planning horizon $[0, H]$ into p_{ij} time zones. For each time zone, the travel speed is constant on arc (i, j). We denote by $T_{ij}^0, T_{ij}^1, ..., T_{ij}^{p_{ij}}$ the time zone boundaries with $T_{ij}^0 = 0$ and $T_{ij}^{p_{ij}} = H$, and by s_{ij}^k the travel speed on time zone $[T_{ij}^k, T_{ij}^{k+1}[$. T_{ij}^k are also called *speed breakpoints* because speed changes occur at these points.

Based on its speed profile, the travel time function $\tau_{ij}(t)$ on an arc (i, j) can be calculated using the procedure described in Ichoua et al. [23]. The main idea in this procedure is that the travel speed is not necessarily constant over the entire length of the road segment, as it changes when the boundary between two consecutive periods is crossed. The resulting travel time function is a continuous piecewise linear function and can be represented by the coordinates of its *travel time breakpoints*. As shown in [23], such travel time function satisfies the FIFO property, i.e., a later departure at node i results in a later arrival at node j.

The quantity $a_{ij}(t) = t + \tau_{ij}(t)$ gives the arrival time at node j given a departure time t at node i. $a_{ij}(t)$ is a continuous piecewise linear non-decreasing function and can be described using coordinates at *arrival time breakpoints* which are directly deduced from *travel time breakpoints*. In this paper, we use arrival time functions as it leads to simpler formulations. Figure 2 illustrates with a simple example inspired from Ichoua et al. [23], the speed, travel time and arrival time functions.
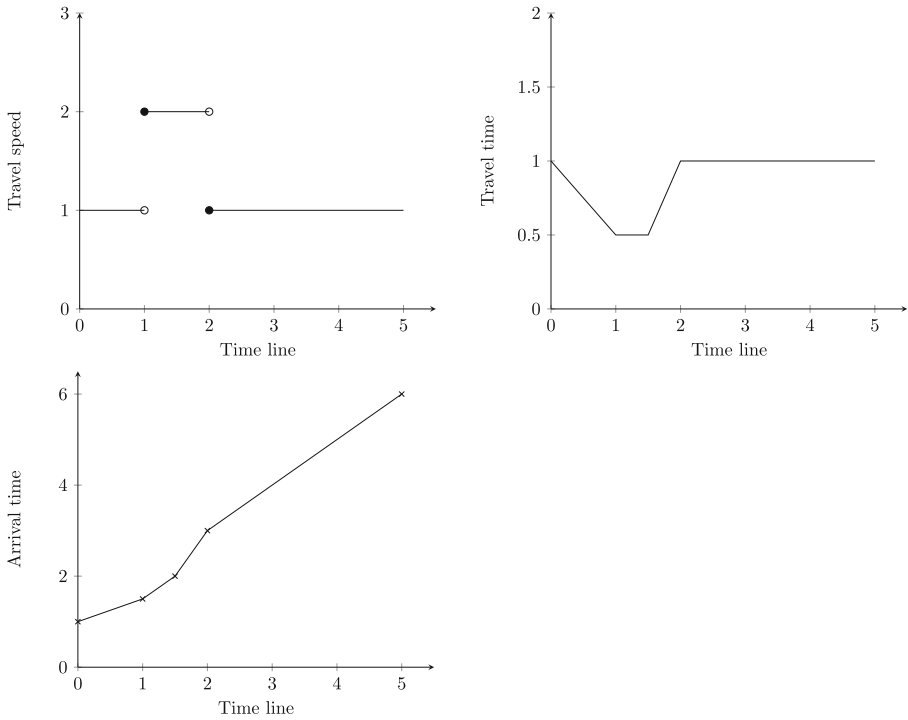
Fig. 2 Speed, travel and arrival times functions for a given arc, with three time zones on interval [0,5]

4 Construction of Customer-Based Graphs

In this section, we focus on algorithms that enable computing the min-cost and min-time graphs. We first present basic notation and operators (Section 4.1). We then report the two algorithms, in Sections 4.2 and 4.3.

4.1 Basic Notation and Operators

Let f be a continuous piecewise linear function, defined on an interval $[0, T]$. f can be described using the following information:

- $N(f)$: number of linear pieces into which f is divided.
- t_f^k: right boundary of the time interval on which is defined the kth piece of f (with also $t_f^0 = 0$).
- $\alpha(f, k)$: linear coefficient of the kth piece of f.
- $\beta(f, k)$: constant term in the kth piece of f.

Using this notation, f can be defined by:

$$f(t) = \begin{cases} \alpha(f,1)t + \beta(f,1) & if \ t_f^0 \leq t \leq t_f^1 \\ \alpha(f,2)t + \beta(f,2) & if \ t_f^1 \leq t \leq t_f^2 \\ \dots & \dots \\ \alpha(f,N(f))t + \beta(f,N(f)) & if \ t_f^{N(f)-1} \leq t \leq t_f^{N(f)} \end{cases}$$

Alternatively, f can be represented using the set of interpolation points

$$I(f) = \{(t_f^0, w_f^0), (t_f^1, w_f^1), \dots, (t_f^{N(f)}, w_f^{N(f)})\}$$

where $w_f^k = f(t_f^k)$.

Let us now consider two continuous piecewise linear functions f_1 and f_2 defined on the same interval $[0, T]$. We say $f_1 < f_2$ if $f_1(t) < f_2(t)$ for all $t \in [0, T]$. Similarly, we say $f_1 \leq f_2$ if $f_1(t) \leq f_2(t)$ for all $t \in [0, T]$. Finally, we define function $min\{f_1, f_2\}$ such that $min\{f_1, f_2\}(t) = min\{f_1(t), f_2(t)\}$ for all $t \in [0, T]$ (see Fig. 3).

Let us now consider four continuous piecewise linear functions f_1, f_2, g_1, g_2 defined on the same interval $[0, T]$. We introduce a function g defined as follows: $g(t) = g_1(t)$ when $f_1(t) < f_2(t)$, $g(t) = g_2(t)$ when $f_2(t) < f_1(t)$, $g(t) = min\{g_1(t), g_2(t)\}$ when $f_1(t) = f_2(t)$ (see Fig. 4, with functions f_1 and f_2 from Fig. 3). We denote this function $\underset{min\{f_1,f_2\}}{merge} \{g_1, g_2\}$.

This last function is used to compute the travel distance function associated with a travel time function. If f_1 and f_2 represent the travel time functions for two different paths connecting the same nodes i and j, and g_1 and g_2 represent the length of these paths, $min\{f_1, f_2\}$ indicates the best travel time that can be obtained from these two paths and $\underset{min\{f_1,f_2\}}{merge} \{g_1, g_2\}$ the associated travel distance.

Let us finally recall how arrival time functions can be composed. Let f_1 and f_2 be arrival time functions for two successive arcs. The composition of f_1 and f_2, denoted
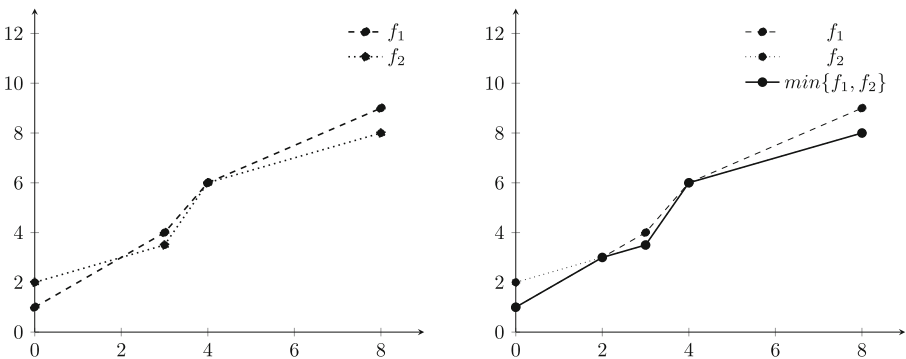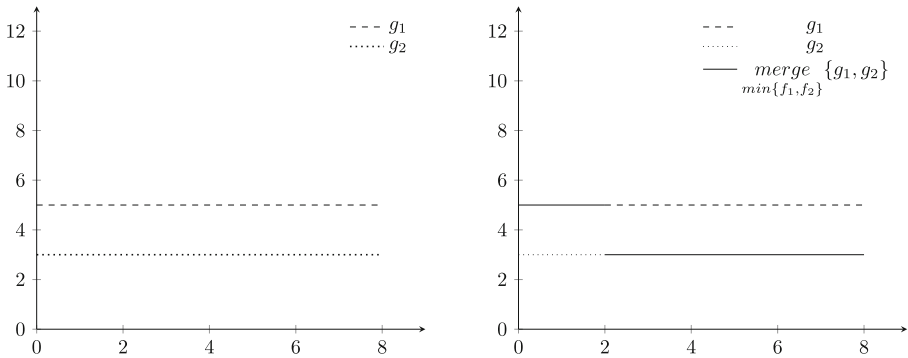

Fig. 3 Function $min\{f_1, f_2\}$

Fig. 4 Function $\underset{min\{f_1,f_2\}}{merge}\ \{g_1, g_2\}$

by $f_2 \circ f_1$, gives the arrival time at the end of the second arc in function of the starting time at the beginning of the first arc. Function $f_2 \circ f_1$ is continuous piecewise linear and has the following set of interpolation points $I(f_2 \circ f_1)$:

- With each interpolation point $(t_{f_1}^k, w_{f_1}^k)$ of f_1 is associated an interpolation point $(t_{f_1}^k, f_2(w_{f_1}^k))$ for $f_2 \circ f_1$.
- With each interpolation point $(t_{f_2}^l, w_{f_2}^l)$ of f_2 is associated an interpolation point $(t_{f_1^{-1}}^l, w_{f_2}^l)$ for $f_2 \circ f_1$ where $t_{f_1^{-1}}^l$ is such that $f_1(t_{f_1^{-1}}^l) = t_{f_2}^l$. Note that $t_{f_1^{-1}}^l$ is easily computed once the index k such that $w_{f_1}^k \leq t_{f_2}^l < w_{f_1}^{k+1}$ is found:
$$t_{f_1^{-1}}^1 = \frac{t_{f_2}^l - \beta(f_1, k)}{\alpha(f_1, k)}.$$

In the next two sections, we present the two algorithms used to compute the min-cost (customer-based) graph and the min-time (customer-based) graph.

4.2 Time-Dependent Shortest Path Algorithm

In the min-cost graph, arcs represent shortest paths (in distance) between points of interest (depot and customer locations). Distance is not time-dependent. All these paths can thus easily be computed by applying $|C| + 1$ times Dijkstra's algorithm in G^{RN}, successively starting from the depot and every customer node.

However, for every pair of points of interest, we also need to determine the arrival time function. Given a min-cost path, the associated arrival time function can be computed starting with the function associated with the first arc, composing it with the function of the second arc, and repeating that with the next arc until the end of the path. Interpolation points for every intermediate function are computed as explained in the previous section.

At first sight, these computations could be carried out after having applied Dijkstra's algorithm. However, it would not function properly because it would not permit to optimize arrival time functions in case of paths with the same distance. In

our implementation, we evaluate the composition every time a label is updated in Dijkstra's algorithm. Details are shown with Algorithm 1.

Algorithm 1 Algorithm for computing time-dependent shortest paths.

1: $d_{ss}^{SP} \leftarrow 0$
2: $a_{ss}^{SP} \leftarrow \hat{0}$
3: **for all** $i \in V^{RN} \setminus \{s\}$ **do**
4:      $d_{si}^{SP} \leftarrow \infty$ and $a_{si}^{SP} \leftarrow \widehat{\infty}$
5: **end for**
6: $Q \leftarrow V^{RN}$ //priority queue of nodes sorted according to d_{si}^{SP}
7: **while** Q is not empty **do**
8:      $i \leftarrow Q.extractMin()$
9:      **for all** $j \in V^{RN}$ such that $(i, j) \in A^{RN}$ **do**
10:          $d \leftarrow d_{si}^{SP} + d_{ij}$
11:          **if** $d < d_{sj}^{SP}$ **then**
12:              $d_{sj}^{SP} \leftarrow d$
13:              $a_{sj}^{SP} \leftarrow a_{ij} \circ a_{si}^{SP}$
14:              $Q.updatePriority(j, d_{sj}^{SP})$
15:          **else if** $d = d_{sj}^{SP}$ **then**
16:              $a_{sj}^{SP} \leftarrow min\{a_{ij} \circ a_{si}^{SP}, a_{sj}^{SP}\}$
17:          **end if**
18:      **end for**
19: **end while**
20: **return** d_{si}^{SP} and a_{si}^{SP} for all $i \in C \cup \{0\}$

The best subpath from the source node $s \in \{0\} \cup C$ to a node $i \in V^{RN}$ is labeled with two values d_{si}^{SP} and a_{si}^{SP} where:

- d_{si}^{SP} represents the length of the subpath;
- a_{si}^{SP} represents the arrival time function at node i, starting at s and following this subpath.

Algorithm 1 is identical to Dijkstra's algorithm except for lines 2, 4, 13 and 15–17. Line 2 initializes to $\hat{0}$ the arrival time function at the source node, where $\hat{0}(t) = 0$, $\forall t \in [0, H]$. Line 4 initializes to $\widehat{\infty}$ the arrival time function for all other nodes, where $\widehat{\infty}(t) = +\infty, \forall t \in [0, H]$. Line 13 updates the arrival time function when a label is modified, i.e., a subpath has been improved. As explained above, the new function is computed by the composition of the function associated with the preceding node and the function associated with the new arc. In case of tie in distance, Lines 15–17 update the arrival time function: the minimal time is kept at each instant.

4.3 Time-Dependent Fastest Path Algorithm

In the min-time graph, arcs represent the fastest paths between points of interest (depot and customer locations). They are described by two functions: an arrival time

function and a distance function. A distance function is needed because the fastest path may change depending on the departure time.

The paths can be computed by applying $|C|+1$ times a labeling algorithm in G^{RN}, providing for a given source node in $s \in C \cup \{0\}$ the fastest paths to all destination nodes in G^{RN} for all possible departure times. This algorithm is depicted in Algorithm 2. It is based on a label correcting procedure where node labels are defined by two functions: the arrival time function a_{si}^{FP} and the distance function d_{si}^{FP}.

These functions are computed using the operators presented in Section 4.1. The main idea of the algorithm is that if the arrival time function f obtained by extending the subpath at a node i along an arc (i, j) improves the arrival time function at node j for at least one $t \in [0, H]$, i.e., there exists $t \in [0, H]$ such that $f(t) < a_{sj}^{FP}(t)$, then the algorithm updates a_{sj}^{FP} to $min\{f, a_{sj}^{FP}\}$. Furthermore, as it means that at that time t a different path is preferred, d_{sj}^{FP} is modified to $\underset{min\{f,a_{sj}^{FP}\}}{merge} \{d_{si}^{FP} + d_{ij}, d_{sj}^{FP}\}$.

Algorithm 2 Algorithm for computing time-dependent fastest paths.

1: $d_{ss}^{FP} \leftarrow \hat{0}$
2: $a_{ss}^{FP} \leftarrow \hat{0}$
3: **for all** $i \in V^{RN} \setminus \{s\}$ **do**
4:      $d_{si}^{FP} \leftarrow \widehat{\infty}$ and $a_{si}^{FP} \leftarrow \widehat{\infty}$
5: **end for**
6: $Q \leftarrow V^{RN}$ // priority queue of nodes sorted according to $a_{si}^{FP}(0)$
7: **while** Q is not empty **do**
8:      $i \leftarrow Q.extractMin()$
9:      **for all** $j \in V^{RN}$ such that $(i, j) \in A^{RN}$ **do**
10:          $f \leftarrow a_{ij} \circ a_{si}^{FP}$
11:          **if not** $a_{sj}^{FP} \leq f$ **then**
12:              $a_{sj}^{FP} \leftarrow min\{f, a_{sj}^{FP}\}$
13:              $d_{sj}^{FP} \leftarrow \underset{min\{f,a_{sj}^{FP}\}}{merge} \{d_{si}^{FP} + d_{ij}, d_{sj}^{FP}\}$
14:              **if** $j \in Q$ **then**
15:                  $Q.updatePriority(j, a_{sj}^{FP}(0))$
16:              **else**
17:                  $Q.insert(j, a_{sj}^{FP}(0))$
18:              **end if**
19:          **end if**
20:      **end for**
21: **end while**
22: **return** d_{si}^{FP} and a_{si}^{FP} for all $i \in C \cup \{0\}$

The structure of this algorithm seems very similar to Algorithm 1. However, there is a huge difference in the management of the priority queue. In Algorithm 1, vertices are ranked according to their distance to the source and can never reenter the queue. In Algorithm 2, there is no total order between arrival time functions. A function

a_{si}^{FP} can be modified after node i has been extracted from the queue, which requires reinserting i in the queue.

5 Branch-and-Price Algorithm for the TDVRPTW$_{RN}$

In this section, we present the branch-and-price scheme used to solve the TDVRPTW$_{RN}$. The algorithm proposed in this paper is based on the branch-and-price scheme presented in Ben Ticha et al. [5] for the same problem with constant travel times. For this reason, we briefly describe the main components of the algorithm. Compared to [5], two main modifications are carried out. First, time-dependent travel information is considered in the pricing problem when extending labels. Secondly, bi-directional dynamic programming is implemented.

The branch-and-price algorithm relies on a master problem that involves a large number of variables and that cannot be solved using a standard branch-and-bound procedure. To handle this issue, a column generation procedure is applied, resulting in a branch-and-price algorithm.

5.1 Master Problem

The master problem for the TDVRPTW$_{RN}$ is given by set covering formulation (1)–(4):

$$Min \sum_{r \in \Omega} c_r x_r \tag{1}$$

$$s.t \ \sum_{r \in \Omega} a_{ir} x_r \geq 1 \ \forall i \in C \tag{2}$$

$$\sum_{r \in \Omega} x_r \leq K \tag{3}$$

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

where Ω denotes the set of feasible routes, c_r is the cost (total traveled distance) of route $r \in \Omega$ and $a_{ir} = 1$ if customer $i \in C$ is served in route r, 0 otherwise. Binary variable x_r takes value 1 if route r is selected in the solution, 0 otherwise.

In the column generation procedure, a restriction of the LP relaxation of Eqs. 1–4 to a subset of routes $\Omega_1 \subset \Omega$, the so-called restricted Master Problem ($MP(\Omega_1)$), is solved at each iteration. Ω_1 is composed of all the routes generated at previous iterations and is iteratively enlarged by solving the pricing problem. For a detailed tutorial on branch-and-price methods, we refer the reader to Feillet [11].

5.2 Pricing Problem

The pricing problem aims at finding new routes offering better ways to serve customers, i.e., with a negative reduced cost. The pricing problem can be reduced to a Time Dependent Shortest Path Problem with Resource Constraints (TDSPPRC) in

G^{RN}. Ben Ticha et al. [5] notice that, in the road-network graph, the elementary path condition has to be modified. Indeed, the set of feasible routes contain routes in which some nodes or arcs of the road network can be traversed several times. In what follows, we say that a route or a path is elementary if customer nodes in the route are served at most once, even if they are traversed several times.

To solve the TDSPPRC, we develop a time-dependent labeling algorithm which is a modification of the labeling algorithm proposed in [5]. This algorithm works as follows. A subpath from the depot to a node $i \in V^{RN}$ is represented using a label defined by $L = (i, t, c, q, S)$ where t is the arrival time at node i (starting from the depot at time 0), c is the reduced cost associated with the subpath, q is the total demand of the customers served along the subpath and S represents the set of customers that are not reachable anymore, either because they have already been served or because they cannot be reached due to resource constraints. Remember that our objective function is to minimize the traveled distance. For that reason, it is always optimal to start from the depot at time 0, which explains that we do not have to maintain a travel time function in labels. Addressing the variant of the problem introduced in Gmira et al. [19], with a travel time minimization objective, would strongly complicate the pricing.

At each iteration, all labels at a certain node $i \in V^{RN}$ are extended along arcs $(i, j) \in A^{RN}$. When the destination node j is a customer node, two extensions are processed. In the first extension, the service at the customer j is performed, if it is feasible. In the second extension, the node is only visited without servicing the associated customer. To handle the exponentially growing number of generated labels, we use the dominance rule proposed by Feillet et al. [12] and defined as follows:

Definition 1 A label $L_1 = (i, t_1, c_1, q_1, S_1)$ dominates a label $L_2 = (i, t_2, c_2, q_2, S_2)$ if:

1. $t_1 \leq t_2$
2. $c_1 \leq c_2$
3. $q_1 \leq q_2$
4. $S_1 \subseteq S_2$

To improve the efficiency of the pricing algorithm, we implement a bi-directional search strategy. To do this, we associate two types of labels with each node i: forward labels defined as described above and backward labels representing subpaths from i to the depot. Backward labels are defined by $L_B = (i, t, c, q, S)$ where t is the latest starting time from node i (enabling reaching the depot before time H), c is the reduced cost, q the total demand and S the set of unreachable customers. Extension for a backward label consists in adding an arc at the beginning of the subpath. New forward labels are only accepted when $t \leq H/2$. Backward labels with $t \leq H/2$ are accepted but are not extended further. Once no more extensions are possible, forward labels and backward labels assigned to the same node are merged to generate complete routes.

Note that the efficiency of the labeling algorithm depends on the length of the subpaths defined by the labels. The bi-directional search strategy aims at limiting this length. More details on bi-directional search can be found in Righini and Salani [32] and Dabia et al. [7].

5.3 Branching Scheme

The branching rule aims at eliminating the current fractional solution by adding constraints and partitioning the solution space. As shown in [5], the standard branching rule used for vehicle routing problems is not suitable in a road-network graph. The reason is that in a customer-based graph, every arc is traversed at most once in a feasible solution. This property does not hold anymore in a road-network graph. We apply the procedure proposed by Ben Ticha et al. [5]:

- We select an arc $(i, j) \in A^{RN}$ with a fractional flow $\phi_{ij} = \sum_{r \in \Omega_1} b_{ijr} x_r$, where b_{ijr} represents the number of times arc (i, j) is traversed along route r;
- We derive two branches:

  - In the first branch, we set the flow upper limit on arc (i, j) to $\lfloor \phi_{ij} \rfloor$
  - In the second branch, we set the flow lower limit on arc (i, j) to $\lfloor \phi_{ij} \rfloor + 1$

To address this branching rule in the column generation procedure, we add constraints (5) or (6) to the restricted master problems:

$$\sum_{r \in \Omega_1} b_{ijr} x_r \leq \lfloor \phi_{ij} \rfloor \qquad \text{with associated dual variable } \lambda_{ij}^{low} \leq 0 \qquad (5)$$

$$\sum_{r \in \Omega_1} b_{ijr} x_r \geq \lfloor \phi_{ij} \rfloor + 1 \qquad \text{with associated dual variable } \lambda_{ij}^{up} \geq 0 \qquad (6)$$

For the pricing, the new dual variable is subtracted from the cost associated with arc (i, j).

Unfortunately, this simple branching rule is not sufficient. Indeed, it can happen that all flow values ϕ_{ij} are integer while the solution is fractional. In this case, we follow Ben Ticha et al. [5] and apply an alternative branching mechanism. Again, we generate two branches:

- In the first branch, we check if graph $\tilde{G} = (V^{RN}, \tilde{A})$, where $\tilde{A} = \{(i, j) \in A^{RN} : \sum_{r \in \Omega_1} b_{ijr} x_r > 0\}$, contains a feasible solution. This is done by first enumerating all the feasible routes in $\tilde{G}$, then solving the resulting set covering formulation with an integer programming solver.
- In the second branch, we enforce the use of at least one arc that is not in $\tilde{A}$. For this aim, we add constraint $\sum_{(i,j) \in A^{RN} \setminus \tilde{A}} \sum_{r \in \Omega_1} b_{ijr} x_r \geq 1$ to the restricted master problem and we subtract the associated dual variable from the cost of every arc $(i, j) \in A^{RN} \setminus \tilde{A}$ when solving the pricing problem.

6 Computational Experiments

In this section, we present our experimental study. We first describe the benchmark problems that we use. Then, we summarize the computational results and we evaluate the advantages of the road-network graph against the two customer-based graphs.

The branch-and-price algorithm is implemented in C++ and we use CPLEX 12.6 for the linear programs. Tests are run on an Intel CORE i5 2.6 GHz computer with 8GB of memory. Computing times for the branch-and-price algorithms with the different graphs are limited to 7200 seconds.

6.1 Test Data

Our experiments are based and a set of instances that simulate the topology of road networks and that we call NEWLET instances. These instances are generated using the procedure proposed in Letchford et al. [29] and that performs as follows. First, it randomly inserts nodes in the Euclidean space, then it considers all possible arcs and inserts a new arc in A^{RN} when two conditions are met: it does not provoke any intersection with other arcs and angles with other arcs at its endpoints are large enough. Arc costs are set according to Euclidean distances.

With this procedure, we generate three sparse graphs with $|V^{RN}| \in \{50, 100, 200\}$. For each sparse graph, three different sets of static travel times with different levels of correlation are computed using formula $t_{ij} = \nu * c_{ij} + \mu * \gamma_{ij} * \bar{c}$ where $\bar{c} = max\{c_{ij} : (i, j) \in A^{RN}\}$, γ_{ij} is a random number in [0, 1] and, ν and μ are correlation parameters defined in [0, 1]. The first set of static travel times has non-correlated travel times (NC), the second set has weakly correlated travel times (WC), the third set has strongly correlated travel times (SC). These travel times represent "nominal" travel times in the road network during non-congested periods and are used to compute "nominal" travel speeds.

To take into account traffic congestion, we associate with every road segment a road profile. We then divide the planning horizon [0, H] into 5 periods: $z_1 = [0, 0.2H[$, $z_2 = [0.2H, 0.3H[$, $z_3 = [0.3H, 0.7H[$, $z_4 = [0.7H, 0.8H[$ and, $z_5 = [0.8H, H[$. The second and fourth periods are congestion periods. Speeds in the remaining periods are relatively high. Implicitly, with this decomposition, we assume that for all road segments, speed breakpoints are the same. It makes sense in practice since congestion tends to happen around the same periods (peak hours) in all the road-network. We assume that there are three types of road segments: congestion-bound, normal, and congestion-free. The type of the road segment is randomly defined and remains the same for the different correlation levels on each road network. Speed on a road segment is then obtained by multiplying its nominal speed by a factor depending on both the segment type and the period, as reported in Table 1.

Given this data, instances are constructed by randomly assigning the depot and the customers to some nodes. For the sparse graph with $|V^{RN}| = 50$, we generate instances with $|C| = 16$ and $|C| = 33$. For the sparse graph with $|V^{RN}| = 100$, we generate instances with $|C| = 25$, $|C| = 33$ and, $|C| = 50$. For the sparse graph with $|V^{RN}| = 200$, we generate instances with $|C| = 25$ and $|C| = 50$. It gives a total of 21 instances.

Table 1 Speed factor profiles

	z_1	z_2	z_3	z_4	z_5
Congestion-free road segments	1.5	1	1.67	1.17	1.33
Normal road segments	1.17	0.67	1.33	0.83	1
Congestion-bound road segments	1	0.33	0.67	0.5	0.83

Each instance is duplicated by assigning either narrow (NTW) or wide time-windows (WTW) to customers. Time windows are defined such that a set of routes, constructed with a greedy algorithm and without considering congestion, is feasible. Finally, an integer service time in {1, 2} is defined for each customer. We set vehicle capacity to 200 and we consider a fleet with a large number of vehicles. We define customer demands such that the routes introduced when defining time windows are feasible.

Table 2 Computing times for the construction of customer-based graphs

$|V^{RN}|$	$|A^{RN}|$	$|C|$	Corr	construction min-cost(s)	construction min-time(s)
50	134	16	NC	0.2	2.0
			WC	0.2	1.5
			SC	0.2	2.1
		33	NC	0.4	4.5
			WC	0.4	4.0
			SC	0.4	4.9
100	286	25	NC	0.5	10.8
			WC	0.5	10.1
			SC	0.5	14.7
		33	NC	0.8	17.3
			WC	0.8	16.1
			SC	0.8	22.5
		50	NC	1.4	29.1
			WC	1.5	26.8
			SC	1.4	34.9
200	580	25	NC	1.3	46.7
			WC	1.3	46.9
			SC	1.3	80.4
		50	NC	3.4	138.7
			WC	3.4	113.0
			SC	3.6	152.1

The 42 instances can be downloaded at the following address: http://absi.nabil.free.fr/#instances.

6.2 Results and Analyses

With our experiments, we evaluate the tractability of the solution of the $TDVRPTW_{RN}$ and we compare solution values with those obtained using customer-based graphs. Table 2 presents the computing times required by the construction of the min-cost and min-time graphs. Tables 3 and 4 compare computing times and solution+ values depending on the graph.

Columns "$|V^{RN}|$", "$|A^{RN}|$" indicate the number of nodes and the number of arcs in the road-network graph. Column "$|C|$" indicates the number of customers. Column "$Corr$" gives the correlation degree between travel times and costs. Columns "*construction min-cost graph(s)*" and "*construction min-time graph(s)*" report the

Table 3 Computing times and solution values for instances with narrow time windows

$\|V^{RN}\|$	$\|A^{RN}\|$	$\|C\|$	Corr	CPU min-cost(s)	CPU min-time(s)	CPU road-network(s)	gap min-cost(%)	gap min-time(%)
50	134	16	NC	**Infeasible**	3.8	95.6	–	**−19.8**
			WC	10.9	4.6	1.7	0.0	**−9.6**
			SC	1.2	0.8	1.0	0.0	**−1.3**
		33	NC	1.2	31.9	3.5	–	**−14.2**
			WC	698.0	672.9	14.4	0.0	**−9.6**
			SC	111.7	60.8	8.9	0.0	**−0.9**
100	286	25	NC	**Infeasible**	0.2	1.1	–	**−7.9**
			WC	0.4	0.2	1.4	0.0	**−8.9**
			SC	0.7	0.4	1.5	0.0	**−4.0**
		33	NC	**Infeasible**	0.2	1.0	–	**−13.3**
			WC	1.0	0.6	2.0	0.0	**−7.6**
			SC	15.9	14.3	49.6	0.0	**−1.3**
		50	NC	4.3	1.8	4.0	0.0	**−6.2**
			WC	26.9	18.5	4.0	0.0	**−3.3**
			SC	199.8	103.5	201.3	0.0	**−3.5**
200	580	25	NC	0.3	0.3	4.0	**−7.8**	**−17.2**
			WC	0.5	0.4	4.0	**−9.1**	**−2.2**
			SC	3.2	2.7	3.9	0.0	**−3.8**
		50	NC	**Infeasible**	47.5	13.0	–	**−16.7**
			WC	52.9	52.9	7089.2	**−2.2**	**−7.7**
			SC	1265.4	1447.9	18.7	0.0	**−1.7**

$^{-}$Instances not solved in 7200 seconds

Table 4 Computing times and solution values for instances with wide time windows

$|V^{RN}|$	$|A^{RN}|$	$|C|$	Corr	CPU min-cost(s)	CPU min-time(s)	CPU road-network(s)	gap min-cost(%)	gap min-time(%)
50	134	16	NC	2.0	1.3	15.0	**−12.4**	**−17.2**
			WC	31.4	16.3	2.8	0.0	**−11.3**
			SC	6.4	3.6	1.7	0.0	**−0.2**
		33	NC	**Infeasible**	2893.2	7125.2	–	**−11.4**
			WC	–	–	–	–	–
			SC	–	–	–	–	–
100	286	25	NC	1.4	1.9	28.2	**−5.3**	**−12.0**
			WC	1.1	0.8	2.4	0.0	**−4.6**
			SC	8.5	4.0	26.2	0.0	**−1.8**
		33	NC	9.5	1.3	2229.9	**−5.6**	**−9.0**
			WC	5.2	6.4	5.1	**−0.2**	**−3.5**
			SC	97.4	45.6	24.4	0.0	**−1.9**
		50	NC	22.2	10.2	6.1	0.0	**−6.1**
			WC	3361.3	2512.8	56.7	0.0	**−3.5**
			SC	–	–	1532.5	–	–
200	580	25	NC	0.9	15.5	7199.2	**−2.0**	**−15.8**
			WC	19.3	2.6	96.6	**−4.7**	**−1.5**
			SC	146.7	119.2	27.4	0.0	**−1.4**
		50	NC	1682.4	5150.0	–	–	–
			WC	4066.2	–	4364.3	**−3.2**	–
			SC	–	–	–	–	–

$^{-}$Instances not solved in 7200 seconds

computing times (in seconds) for the construction of the customer-based graphs. Columns "*CPU min-cost(s)*", "*CPU min-time(s)*" and "*CPU road-network(s)*" report the computing times of the branch-and-price algorithms, expressed in seconds. The "gap" columns report the gap between solution values of the TDVRPTW_{RN} and solution values in the associated customer-based graph. These gaps are computed as follows:

$$gap(\%) = \frac{costs\ in\ road\ network\ graph - costs\ in\ customer\ based\ graph}{costs\ in\ customer\ based\ graph} * 100 \tag{7}$$

As expected, we see from Table 2 that the computing time needed for the construction of the min-time graph is larger than that of the min-cost graph, and that it increases with the number of customers.

From Tables 3 and 4, it comes out that 38 out of 42 TDVRPTW_{RN} instances are solved, while 32 and 37 optimal solutions only are obtained when using the min-cost

and min-time graphs, respectively (plus 5 proven infeasibility for the min-cost graph). Computing times are however very unpredictable. In some instances (see, e.g., the last two instances in Table 3), huge differences can appear between the three solution methods, not always at the advantage of the same method. Precisely, the branch-and-price algorithm on the road-network graph is faster for 14 out of 31 solved instances compared to the min-cost graph and is faster for 14 out of 36 solved instances compared to the min-time graph. The interesting point is that on these instances using the road-network is tractable, at least as much as using customer-based graphs.

The fact that the efficiency of branch-and-price algorithms is very dependent on instance characteristics is known for a long time for vehicle routing problems. It is generally admitted that it depends on the number of nodes in the routes that are generated by the pricing problem. It is more surprising and difficult to explain that the same instance can be solved so irregularly depending on the implementation. Looking carefully at Tables 3 and 4, a correlation can be observed between computing times obtained with the two customer-based graphs. Probably, the very different behavior observed when the road-network is used, is due to the fact that the branch-and-price is operated on a very different network. The number of nodes in routes found in this network is not strictly correlated to the number of visited customers, that is, the number of nodes of equivalent routes in the customer-based graphs.

Regarding solution values, the $TDVRPTW_{RN}$ enables to achieve important improvements. Solution costs are reduced for 10 instances compared to solutions obtained with the min-cost graph and for 36 instances compared to solutions obtained with the min-time graph. Gaps are up to 12.4% and 19.8%, respectively. Average gaps are 1.7% and 7.3%. Furthermore, the min-cost graph does not admit any feasible solution for 5 instances (due to the loss in the graph of more costly but faster paths).

Another interesting observation is that increasing the correlation between costs and travel times reduces the improvements in solution costs obtained when conserving the complete road-network information. Especially, gaps are most of the time significantly larger for NC instances. While strong correlations can be expected between travel times and distances, it means that road-network information is specially relevant when compromises between less correlated attributes are sought, for example, between travel time and danger for hazmat transportation.

7 Conclusions

In this paper, we investigated the Time-Dependent Vehicle Routing Problem with Time Windows and Road-Network Information ($TDVRPTW_{RN}$). The originality of this model is that time-dependent travel time information is defined on road-network arcs. It goes against the traditional modeling paradigm of time-dependent vehicle routing problems where a customer-based graph is first introduced and travel time functions are then directly defined on this graph arcs. We only found a few papers following the same line in the literature.

The TDVRPTW was selected for this study as an exemplary problem involving travel time and another attribute, namely travel distance. We proposed an exact solution approach based on branch-and-price. As far as we know, this is the first

work addressing the exact solution of a time-dependent VRP starting from input data defined at the road-network level. Previous works either consider a limited set of options for traveling between customers (e.g., Huang et al. [22]) or develop heuristic solution schemes (Gmira et al. [19]).

With our experiments, we demonstrated that important savings could be obtained by working on the road-network graph compared to two alternative approaches: using the min-cost graph or the min-time graph. Regarding the min-time graph, large gaps, up to almost 20%, could be obtained on solution costs, especially when the two attributes (travel-time and cost) are assumed not to be correlated. With regards to the min-cost graph, many false infeasible situations were observed: no solution was found with the min-cost graph while the problem is feasible.

Several important algorithms are also presented in this paper. First, we describe two original dynamic programming algorithms to compute the travel-time and travel distance functions in the min-cost and min-time graphs. This is new in the time-dependent VRP literature where such algorithms cannot be found. Second, we adapt a branch-and-price scheme to the TDVRPTW$_{RN}$ to be able to derive optimal solutions while considering complete road-network information. The main specificity of this algorithm is to apply branch-and-price directly on the road-network graph.

This algorithm and the two branch-and-price algorithms for which vehicle routes are defined on customer-based graphs, show however some limits. Actually, the main purpose of our study was to provide a complete solution scheme, starting from realistic time-dependent input data at the road-network level and computing optimized vehicle routes. Even so, we quickly started checking how the algorithm would scale up. We considered a road-network extracted from **OpenStreetMap©** (www.openstreetmap.org/), with 5437 nodes and 19500 arcs. On this network, we could solve instances with up to 25 customers. Larger instances were out of reach. More details are given in the Appendix. To deal with larger instances, the proposed branch-and-price framework could be enhanced in many ways. For example, the special structure of the road-network graph could be exploited much more in the pricing problem. Also, heuristic pricing methods might be designed. Introducing valid inequalities, in a branch-cut-and-price framework could also be very helpful.

Another interesting research direction that is left for perspective would be to develop multigraphs built upon accurately selected efficient paths (e.g., the min-cost and min-time paths together). It would allow keeping an efficiency similar to that of customer-based graphs with more limited impacts on solution quality.

Compliance with Ethical Standards

Conflict of Interest The authors declare that there is no conflict of interest.

Appendix. Experiments on larger instances

In this Appendix, we report preliminary results on larger instances.

A.1 Instance Generation

The new instances are generated based on real data from the road network of the central urban area of the city of *Aix-en-Provence* (a city-commune in the south of France). Spatial data is extracted from **OpenStreetMap©** (www.openstreetmap.org/). We obtain a road-network graph with 5437 nodes and 19500 arcs (see Fig. 5). Each arc is defined with a length and a maximum allowed speed. Costs are set as road segment lengths.

Time periods and speed profiles are defined as described in Section 6.1. Road segment types are defined according to maximum allowed speeds. For highways, motorways, and arterial roads (characterized with a high maximum allowed speed), the road segment type is set to "normal". For streets, boulevards, and roads in the center of the city, the road segment type is set to "congestion-bound". For small roads and living streets (characterized with a low maximum allowed speed), the road segment type is set to "congestion-free".

Based on this road network, we generate instances with $|C| \in \{5, 10, 25\}$, with three instances for each value of $|C|$. Depot and customer locations, time windows, customer demands, service times, and vehicle capacity are defined in the same way as for NEWLET instances. We call these instances AIX instances.
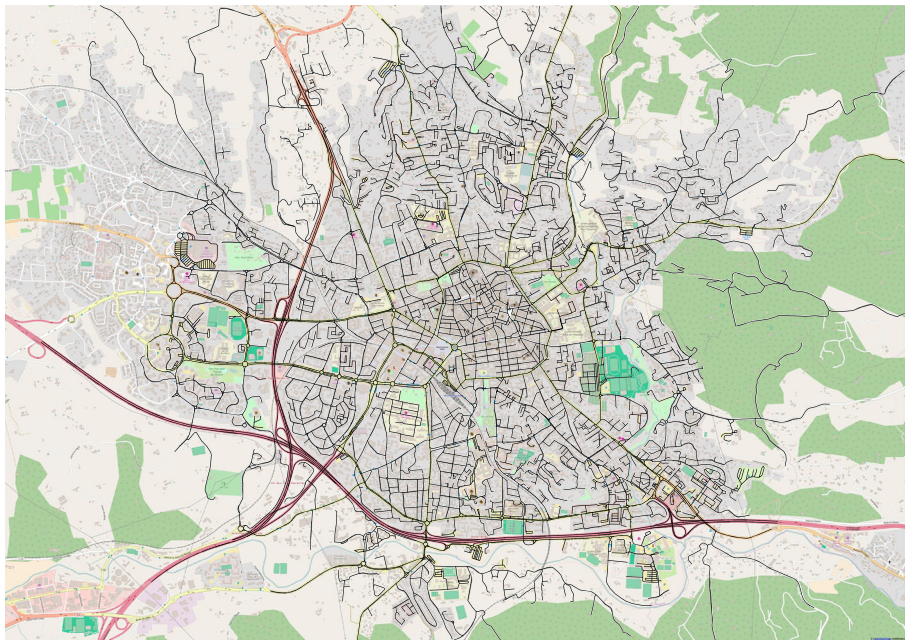

Fig. 5   Road Network of the central urban area of Aix-en-Provence (France)

A.2 Experiments

Table 5 reports the results obtained for AIX instances. Headings are the same as in previous tables, except Column "Ins", which indicates the instance index. Note that results are not reported for min-time graphs. Indeed, due to the complexity of the proposed algorithm (see Section 4.3), we could not generate complete min-time graphs in a reasonable amount of time for this road-network.

Table 5 Computing times and solution values for AIX instances

$|C|$	Ins	construction min-cost(s)	CPU min-cost(s)	CPU road-network(s)	gap min-cost(%)
5	1	12.8	0.02	7.0	**−10.4**
	2	17.8	0.03	8.2	**−11.8**
	3	17.2	0.10	24.4	**−7.6**
10	1	28.3	0.06	18.3	**−7.1**
	2	43.0	0.05	12.7	**−6.9**
	3	34.9	0.05	40.9	**−2.1**
25	1	129.3	0.09	97.9	**−0.6**
	2	114.6	0.09	56.8	**−1.4**
	3	164.3	0.08	35.8	**−4.3**

In these instances, the size of the road-network and the small density of customer nodes in the network are representative of what can be expected in real distribution systems. Especially, the road-network is much larger than customer-based graphs. We observe that solving the TDVRPTW$_{RN}$ becomes more complicated. Only instances with a limited number of customers can be solved in a reasonable amount of time. However, we also observe that the benefits are there. Solving the TDVRPTW$_{RN}$ enables improving solution costs for all the instances, in amounts largely greater than those obtained on NEWLET instances. The saving is 5.8% on average and reaches 11.8%. We also see that the improvement in solution costs decreases when the number of customer nodes increases. The average improvement is 9.9% for instances with 5 customers and goes down to 2.1% for instances with 25 customers.

References

1. Beasley JE (1981) Adapting the savings algorithm for varying inter-customer travel times. Omega 9(6):658–659
2. Ticha HB, Absi N, Feillet D, Quilliot A (2019) Multigraph modeling and adaptive large neighborhood search for the vehicle routing problem with time windows. Comput Oper Res 104:113–126
3. Ticha HB, Absi N, Feillet D, Quilliot A (2017) Empirical analysis for the vrptw with a multigraph representation for the road network. Comput Oper Res 88:103–116

4. Ticha HB, Absi N, Feillet D, information AlainQuilliot. (2018) Vehicle routing problems with road-network State of the art. Networks 72:393–406

5. Ticha HB, Absi N, Feillet D, Quilliot A, Van T (2019) Woensel. A branch-and-price algorithm for the vehicle routing problem with time windows on a road-network graph. Networks 73(4):401–417

6. Cordeau J-F, Ghiani G, Guerriero E (2012) Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. Transp Sci 48(1):46–58

7. Dabia S, Ropke S, Woensel TV, Kok TD (2013) Branch and price for the time-dependent vehicle routing problem with time windows. Transp Sci 47(3):380–396

8. Delling D, Wagner D (2009) Time-dependent route planning. In: Ahuja RK, Möhring RH, Zaroliagis CD (eds) Robust and online large-scale optimization. Lecture Notes in Computer Science, vol 5868. Springer, Berlin

9. Donati AV, Montemanni R, Casagrande N, Rizzoli AE, Gambardella LM (2008) Time dependent vehicle routing problem with a multi ant colony system. Eur J Oper Res 185(3):1174–1191

10. Eglese R, Maden W, Slater A (2006) A road timetable to aid vehicle routing and scheduling. Comput Oper Res 33(12):3508–3519

11. Feillet D (2010) A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR Q J Oper Res 8(4):407–424

12. Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary short-est path problem with resource constraints Application to some vehicle routing problems. Networks 44(3):216–229

13. Figliozzi MA (2012) The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. Transp Res E Logist Transp Rev 48(3):616–636

14. Fleischmann B, Gietz M, Gnutzmann S (2004) Time-varying travel times in vehicle routing. Transp Sci 38(2):160–173

15. Garaix T, Artigues C, Feillet D, paths DidierJosselin. (2010) Vehicle routing problems with alternative An application to on-demand transportation. Eur J Oper Res 204(1):62–75

16. Gendreau M, Ghiani G, Guerriero E (2015) Time-dependent routing problems: a review. Comput Oper Res 64:189–197

17. Ghiani G, Guerriero E (2014) A note on the Ichoua, Gendreau, and Potvin (2003) travel time model. Transp Sci 48(3):458–462

18. Gmira M (2019) Confection de tournées de livraison dans un réseau urbain à l'aide de métaheuristiques et de méthodes de forage de données massives. PhD thesis. Polytechnique Montréal

19. Gmira M, Gendreau M, Lodi A, Potvin J-Y (2020) Tabu search for the time-dependent vehicle routing problem with time windows on a road network. Eur J Oper Res

20. Golden BL, Raghavan S, Wasil EA (2008) The vehicle routing problem: latest advances and new challenges, vol 43. Springer Science & Business Media, New York

21. Hansen P (1980) Bicriterion path problems. In: Multiple criteria decision making theory and application. Springer, pp 109–127

22. Huang Y, Zhao L, Woensel TV, Gross J-P (2017) Time-dependent vehicle routing problem with path flexibility. Transp Res B Methodol 95:169–195

23. Ichoua S, Gendreau M, Potvin J-Y (2003) Vehicle dispatching with time-dependent travel times. Eur J Oper Res 144(2):379–396

24. Jabali O, van Woensel T, de Kok T (2012) Analysis of travel times and CO_2 emissions in time-dependent vehicle routing. Prod Oper Manag 21(6):1060–1074

25. Kaufman DE, Smith RL (1993) Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. J Intell Transp Syst 1(1):1–11

26. Kok AL, Hans EW, Schutten JMJ (2012) Vehicle routing under time-dependent travel times: the impact of congestion avoidance. Comput Oper Res 39(5):910–918

27. Lai DSW, Demirag OC, Leung JMY (2016) A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. Transp Res E Logist Transp Rev 86:32–52

28. Laporte G (2009) Fifty years of vehicle routing. Transp Sci 43(4):408–416

29. Letchford AN, Nasiri SD, Oukil A (2014) Pricing routines for vehicle routing with time windows on road networks. Comput Oper Res 51:331–337

30. Orda A, Rom R (1990) Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. J ACM (JACM) 37(3):607–625

31. Patoghi A, Shakeri Z, Setak M (2017) A time dependent pollution routing problem in multi-graph. Int J Eng 30(2):234–242
32. Righini G, Salani M (2006) Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discret Optim 3(3):255–273
33. Serafini P (1987) Some considerations about computational complexity for multi objective combinatorial problems. In: Recent advances and historical development of vector optimization. Springer, pp 222–232
34. Setak M, Habibi M, Karimi H, Abedzadeh M (2015) A time-dependent vehicle routing problem in multigraph with fifo property. J Manuf Syst 35(35):37–45
35. Tikani H, Setak M (2019) Efficient solution algorithms for a time-critical reliable transportation problem in multigraph networks with fifo property. Appl Soft Comput 74:504–528
36. Toth P, Vigo D (2014) Vehicle routing: problems, methods, and applications. SIAM
37. Wang H-F, Lee Y-Y (2014) Two-stage particle swarm optimization algorithm for the time dependent alternative vehicle routing problem. J Appl Comput Math 3(4):1–9

Affiliations

Hamza Ben Ticha[1] · Nabil Absi[1] · Dominique Feillet[1] ⬤ · Alain Quilliot[2] · Tom Van Woensel[3]

Hamza Ben Ticha
hamza.ben-ticha@emse.fr

Nabil Absi
absi@emse.fr

Alain Quilliot
alain.quilliot@isima.fr

Tom Van Woensel
T.v.Woensel@tue.nl

[1]　Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Centre CMP, F - 13541, Gardanne, France
[2]　LIMOS, Institut Supérieur d'Informatique de Modélisation et leurs Applications, ISIMA, Campus des Cèzeaux, Aubière Cedex, France
[3]　School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands