The Vehicle Routing Problem with Access Restrictions

Munise Kübra Şahin
ORSTAT, Faculty of Economics and Business, KU Leuven, 3000 Leuven, Belgium. munisekubra.sahin@kuleuven.be

Hande Yaman
ORSTAT, Faculty of Economics and Business, KU Leuven, 3000 Leuven, Belgium. hande.yaman@kuleuven.be

To mitigate the negative effect of freight vehicles on urban areas, many cities have implemented road accessibility restrictions, including limited traffic zones, which restrict access to specific areas during certain times of the day. Implementing these zones creates a trade-off between the delivery cost and time, even under the assumption of equal traversal time and travel cost. Consequently, the planners in charge of vehicle routing need to work with graphs containing information on all Pareto-optimal paths. Inspired by these changes in city logistics and the resulting computational challenges, we study the vehicle routing problem with access restrictions, where some streets are closed to traffic within a given time period. We formulate this problem using workday variables and propose two branch and price algorithms based on the underlying road network and multi-graph. The results of our computational experiments demonstrate the effectiveness of the proposed algorithms, solving instances with up to 100 nodes and 33 customers, and underline the importance of considering alternative paths in reducing costs.

Key words: vehicle routing; access restrictions; road network; multi-graph; branch and price; sustainable city logistics

1. Introduction

The inner-city operation of freight vehicles causes congestion, air pollution, and noise pollution, leading to serious public health and environmental issues. To cope with these consequences, municipalities around the world implement road accessibility restrictions, such as *low emission zones* and *limited traffic zones*. This study focuses on the latter, which restricts access to specific areas during certain times of the day. While these restrictions reduce the aforementioned problems, they pose challenges to logistics companies accessing the customers in the restricted areas. The deliveries in question typically occur during business hours. Restricted access in these time intervals can increase travel distance, causing higher costs and longer delivery times and forcing companies to use more vehicles. Despite the challenges that limited traffic zones pose to logistics providers, imposing access restrictions remains an important strategy for creating more sustainable urban environments. Companies should incorporate these restrictions in their logistics planning to decrease the negative impact of these measures on their costs and quality of service.

With this motivation, we define the Vehicle Routing Problem with Access Restrictions (VRPAR). The aim is to plan delivery routes for a homogeneous fleet of vehicles in a city where vehicles are not allowed to enter certain arcs during a given time period. Customers do not quote time windows and can be visited any time during the day, and the demands of customers cannot be split among several vehicles. We assume that vehicles are unlimited in number and can be used by incurring a fixed cost for a workday. Vehicles have capacities, and the duration of a workday limits their working time. The aim is to find minimum cost routes that comply with access restrictions, in which every customer is visited exactly once, the capacity restrictions of the vehicles are respected, and the vehicles can be back at the depot by the end of the workday.

In this study, we assume that the time it takes to traverse an arc is equal to (or a positive multiple of) the cost incurred to traverse the arc, for all arcs. Under this assumption, if there are no access restrictions, then in an optimal solution, one uses only minimum cost routes. Indeed, the classical VRP is defined on a complete graph, where the cost of using an arc (i, j) is equal to the cost of the minimum cost path from i to j. However, in VRPAR, vehicles may encounter different waiting times depending on the arrival time at the tail of an arc with access restrictions. When this waiting time is incorporated into the travel time of the corresponding arc, VRPAR becomes a special case of the time-dependent VRP where some arcs have time-dependent travel times. Consequently, despite the fact that the traversal time of an arc is equal to its travel cost, it is not always possible to consider only minimum cost paths between two nodes. To demonstrate this feature, Figure 1 depicts a graph where the numbers on the arcs are the travel costs (also traversal times). The graph consists of eight road junctions, where no customer or depot is located, and two customers, represented by orange nodes. The restricted arcs closed to the traffic between times 60 and 100 are represented by dashed lines. There are four paths to go from customer 1 to customer 2. The arrival times of all paths are plotted on the right as a function of the dispatch time from customer 1. This plot shows that more expensive paths have shorter travel times for certain dispatch times. For example, the most expensive path, 1-8-9-2, leads to the earliest arrival time for departures between times five and ten.

As this example shows, assuming the equality of traversal times and costs does not suffice to be able to solve the VRPAR in a graph that contains "one best arc" between two nodes. In other words, a graph that contains the information on all Pareto-optimal paths must be considered not to jeopardize optimality. Hence VRPAR belongs to the family of VRPs with alternative paths. We note that the workday restriction is critical in VRPAR, since, without it, the vehicles will wait until the end of the restricted period instead of taking more costly routes, and the problem reduces to the capacitated VRP.

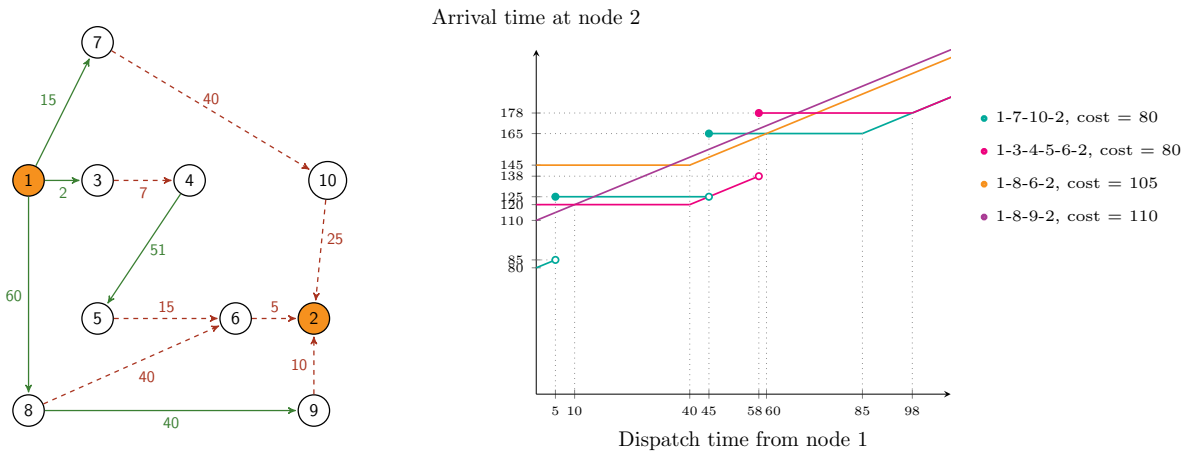


Figure 1 **The effect of access restrictions on the arrival times**

To the best of our knowledge, alternative paths arising from the incorporation of access restrictions have never been considered in the literature. With this study, we aim to fill this gap. We propose two branch and price algorithms to solve VRPAR, one based on the *road network* and the other one based on the *multi-graph*, defined in Sections 4 and 5, respectively. Our computational experiments show that the proposed algorithms can solve instances with up to 100 nodes and 33 customers, and the multi-graph approach performs better on the tested instances.

The rest of the paper is organized as follows. In Section 2, we review the literature. Section 3 provides a formal definition of the VRPAR and introduces our formulation with path-based variables on the road network. We describe the proposed branch and price algorithm based on the road network in Section 4. We explain the construction of the multi-graph and the required changes on the model and the algorithm for the multi-graph approach in Section 5. In Section 6, we present the results of our computational experiments. Finally, we conclude in Section 7.

2. Related literature

To the best of our knowledge, imposing access restrictions has been studied only in the context of the VRP with Access Time Windows (VRPATW), defined by Muñuzuri et al. (2013). In VRPATW, a restricted area that contains a subset of the nodes is closed to the traffic within a given time period. Since the vehicles do not have access to the whole area during the restricted period, the nodes inside this area cannot be visited within the restricted period, no matter which path is used. Therefore, the access time windows in VRPATW do not create Pareto-optimal paths when the arcs have one distinctive attribute as opposed to our setting. As a result, VRPATW is closely related to the VRP with time windows, where the time windows correspond to the intervals during which the nodes cannot be visited. The authors propose a heuristic algorithm where the routes are first generated by a genetic algorithm without considering the access restrictions, and then the

obtained customer sequences are modified in a way that the access restrictions are respected. Later on, Grosso et al. (2018) propose a mathematical formulation for this problem.

In the remainder of this section, we review the results and algorithms for two problems closely related with VRPAR, namely, VRP with alternative paths and time-dependent VRP. To facilitate the discussion, we define three commonly used graphs in the related studies: road network, multi-graph and customer-based graph. In road networks, each arc represents the road segments, and the nodes correspond to the endpoints of these segments. Therefore, the node set might also contain the road junctions where no customer or depot is located. An example of this graph type can be seen in Figure 1. In multi-graphs, the node set contains only the set of depots and the set of customers. The arc set contains all Pareto-optimal paths obtained by solving a multi-criteria shortest path problem in the underlying road network for each node pair. Hence, there might be parallel arcs between each node pair. When only one attribute is considered, and each node pair is linked by at most one arc representing the best path on the underlying road network in terms of this attribute, the resulting graph is called a customer-based graph. One example of a customer-based graph is the min-cost graph, which contains only the arcs representing min-cost paths on the underlying road network. Figure 2 demonstrates the multi-graph and min-cost graph representations of the road segment depicted in Figure 1, where one of the min-cost paths is arbitrarily chosen for the latter.

Figure 2 **Multi-graph and min-cost graph representations of Figure 1**

2.1. VRP with alternative paths

Most of the studies in the VRP literature use customer-based graphs. However, the shortest and quickest paths are not always the same in reality. Therefore, addressing the problem on customer-based graph results in discarding the paths offering trade-offs between different attributes and might lead to overestimating cost or infeasibility. To the best of our knowledge, Garaix et al. (2010) is the first one addressing the disadvantages of customer-based graphs when several attributes are associated with the arcs. The authors propose constructing a multi-graph. They solve a dial-a-ride problem using the branch-and-price algorithm, where the pricing problem is solved over the multi-graph. They also point out that the sequence of visited customers cannot be used to define a route due to potentially having multiple arcs between each node pair. Even when this order is known, the resulting problem is a multiple-choice knapsack problem (multidimensional if more than one

attribute is constrained). Although the authors indicate that they construct the multi-graph using a labelling algorithm similar to the one used in the pricing problem, they leave the space and time requirements of such a graph as an open question. This question is later addressed by Letchford, Nasiri, and Oukil (2014). The authors show that when the road network contains $\mathcal{O}(|J|\log T)$ arcs where J is the set of customers and T is the workday duration limit, the multi-graph can contain $\Theta(|J|^2T)$ arcs. To remedy this problem, they propose using the underlying road network directly, which requires allowing the customers to be visited more than once in the pricing problem: either to serve or to pass through without serving. Whereas the pricing algorithm is explained in detail, the branching part of the algorithm is left for future research. Later, Ben Ticha et al. (2019) complete this work and propose a branch-and-price algorithm for the VRPTWs on a road network. They show that integer arc flows do not suffice for the integrality of the VRPs on a road network since the arcs can be traversed more than once. They propose a hierarchical branching where branching on the arc flows is prioritized. If a fractional solution is detected while all arc flows are integer, they construct a support graph. In the first branch, they enumerate all feasible routes in the support graph and solve the master problem using these enumerated routes. In the second branch, they enforce at least one arc that is not used in the previous solution to be used. They also compare the efficacies of the algorithms on a road network and a multi-graph using their previous paper, Ticha et al. (2017), in which a branch-and-price algorithm for the VRPTWs on a multi-graph is proposed. Their computational results show that the multi-graph approach outperforms the road network approach. However, the authors also note that the size of the multi-graph could significantly increase for some extensions of the VRP and even be intractable. This reason drives the authors towards using the road network for the time-dependent VRPTW, as explained in the next section.

The literature on the VRPs on a road network has also focused on the arc routing problems (ARPs) in which the service is performed at the arcs instead of nodes. Two approaches stand out in the ARP literature, namely, modelling the problem as a node-routing problem after converting the road network into a customer-based one with additional nodes and directly using the road network. For brevity, we do not review the ARP literature and refer interested readers to the recent survey of Corberán et al. (2021).

2.2. Time-dependent VRP

In the traditional VRP, the travel times of the arcs are assumed to be time-independent. However, travel times might be affected by exogenous factors such as traffic congestion or endogenous factors such as the vehicle's speed chosen by the driver. Modelling such situations requires relaxing the assumption of time-independent travel times. The arising problem is called time-dependent VRP (TDVRP).

Dabia et al. (2013) propose the first exact algorithm for TDVRP where they consider the extension with time windows and minimize the total route duration. As the feasibility of a route is defined considering the time dependency of travel times, their master problem is the same as that of VRPTW, but the resulting pricing problem turns into a time-dependent elementary shortest path problem with resource constraints. They solve this problem using a tailored labelling algorithm and branch on the arc variables using strong branching. Since the cost of a route (its duration) depends on the dispatch time from the depot, so is its reduced cost (i.e., it cannot be directly decomposed to each edge). Therefore, in the proposed labelling algorithm, each label stores a piecewise linear function of the dispatch time from the depot to represent the service completion time at the corresponding end node. When a label is completed, among the dispatch times from the depot for which the label is feasible, the one that provides the minimum route duration is selected using this piecewise linear function. Then, the reduced cost of the route is calculated according to the selected dispatch time.

The pollution routing problem (PRP) is an extension of the VRPTW in which travel speeds at each arc must be determined in addition to the vehicle route to minimize the consumption of a vehicle and the cost of a driver Bektaş and Laporte (2011). Whereas the speed on each arc is bounded because of the traffic regulations in the standard PRP, Franceschetti et al. (2013) consider the speed restrictions caused by traffic congestion and employ time-dependent travel times. Incorporating the congestion into the PRP framework opens up a new question that has not been investigated in the VRPTW literature before: under which conditions should idle waiting after a service completion be used as a strategy to avoid congestion? The authors provide a characterization of the optimal solutions for the single-arc version of the problem, which is used to identify the conditions under which idle waiting at certain locations is optimal.

In the studies mentioned above, any two nodes are linked by at most one arc. To the best of our knowledge, Setak et al. (2015) is the first one considering the time-minimizing TDVRP in a multi-graph. They propose a tabu search algorithm to solve this problem. Even though multiple arcs are considered between each node pair, different trade-offs are not induced by the choice of an arc because the travel costs are not considered. In other words, one arc dominates the others once the travel start time is known. It is also worth mentioning that the authors do not discuss the tractability of the multi-graph and restrict their computational experiments to the graphs that contain at most three arcs between each pair of nodes.

Huang et al. (2017) introduce the cost-minimizing TDVRP with path flexibility and limited workday duration. They consider a cost function that consists of the fuel cost dependent on vehicle speed and load and vehicle depreciation cost dependent on the travel distance. Whereas in the time-minimizing TDVRPs without budget constraints, selecting the fastest path of a certain

Table 1 A classification of the studies in the literature

	customer-graph	multi-graph	road network	objective	time-dependent	branching
Garaix et al. (2010)	✗	✓	✗	total cost	✗	enforce or forbid a successor
Dabia et al. (2013)	✓	✗	✗	total route duration	✓	enforce or forbid an arc
Ticha et al. (2017)	✗	✓	✗	total cost	✗	enforce or forbid an arc
Ben Ticha et al. (2019)	✗	✓	✓	total cost	✗	arc flows + support graph
Ben Ticha et al. (2021)	✗	✗	✓	total cost	✓	arc flows + support graph

dispatch time between each pair of nodes is always optimal if travel times satisfy the first-in-first-out condition, locally cost-minimizing paths might not be globally optimal for the considered objective function. Therefore, the authors define a set of candidate paths and provide a mathematical formulation to decide the sequence of the nodes as well as the paths used between two consecutive nodes from among the candidates. A set of candidate paths contains the shortest paths in terms of time for a set of selected dispatch times and the shortest path in terms of distance for each pair of nodes. Recently, Ben Ticha et al. (2021) and Jaballah et al. (2021) extend this problem to the TDVRP on the road network. The former proposes a branch-and-price algorithm and minimizes the total travel distance, and the latter proposes a simulated annealing algorithm and minimizes the total travel time. Minimizing the total travel distance rather than time enables Ben Ticha et al. (2021) to use the same pricing algorithm of Ben Ticha et al. (2019) proposed for the time-independent version by simply changing the computation of the travel times. Even though cost-minimizing objectives allow the standard labelling algorithms to be used in time-dependent problems, the challenge is that they cannot be converted into the TDVRP. On the other hand, as mentioned before, their time-minimizing counterparts can be by solving all pairs of shortest path problem for each possible starting time (i.e, by choosing the locally time-minimizing paths). Even modelling the cost-minimizing TDVRPs subject to the time-related constraints on a multi-graph requires solving multi-criteria all-pairs shortest path problem for each possible starting time.

In Table 1, we provide the classification of the studies that propose an exact algorithm.

3. Problem definition and a set covering formulation on the road network

Let $G = (N, A)$ be a directed graph, representing the road network where set N consists of the depot 0, set of customers J and set of road junctions K at which no customer or depot is located. Each customer $i \in J$ has a nonnegative demand q_i, and each arc $a \in A$ is associated with a nonnegative traversal time t_a and nonnegative travel cost c_a. Arc a starts at node i_a and ends at node j_a. For a subset of arcs A_R, there is an access limitation during times $[e, l)$. Because of this limitation, the

actual travel time on an arc depends on when we reach the tail of the arc. We represent the travel time of arc a where d is the time that the vehicle is ready to travel on arc a as $\tau_a(d)$ and calculate it as follows:

$$\tau_a(d) = \begin{cases} t_a & \text{if } a \notin A_R \vee d < e \vee d \geq l, \\ l - d + t_a & \text{otherwise.} \end{cases} \tag{1}$$

Given a partial path $p = (a_0, \ldots, a_k)$ that starts at node i, let $\delta_a^p(d)$ be the arrival time at node j_a given a dispatch time d at node i. The arrival time at each node visited by the path can be recursively calculated as follows:

$$\delta_{a_i}^p(d) = \delta_{a_{i-1}}^p(d) + \tau_{a_i}\left(\delta_{a_{i-1}}^p(d)\right) \quad \forall i \in \{0, \ldots, k\} \tag{2}$$

where $\delta_{a_{-1}}^p(d) = d$.

The travel time of path p for dispatch time d can be calculated as $\tau^p(d) = \delta_{a_k}^p(d) - d$. We show the travel time and the arrival time graphs of arc $a \in A_R$ in Figure 3.

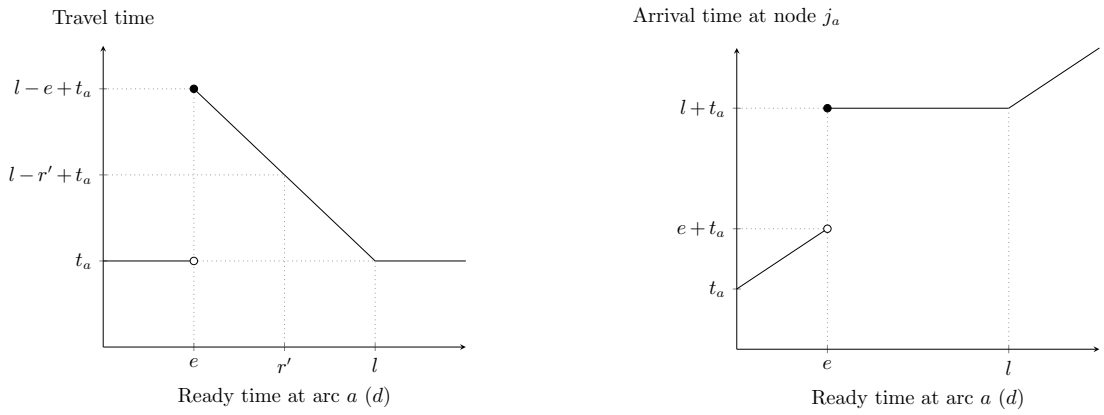


Figure 3 **Travel and arrival time graphs of arc** $a \in A_R$

There are an unlimited number of vehicles with capacity Q and a fixed cost f at the depot. The problem is finding a set of trips that minimizes the total cost while ensuring that each customer is served exactly once and all vehicles are back at the depot by time T. Each trip should also comply with access restrictions and vehicle capacity constraints.

We formulate VRPAR using path variables. We call a path feasible if it starts and ends at the depot, and if it respects the access time windows, workday duration limit and vehicle capacity constraints. We define the set of all feasible paths as P. We represent the set of arcs traversed in path $p \in P$ by A_p, and the cost of path p by $\Delta_p = \sum_{a \in A_p} n_{ap} c_a$ where n_{ap} represents the number of times arc a is traversed by path p. Binary parameter s_p^j is equal to 1 iff customer $j \in J$ is served in

path p. The variable z_p takes value 1 if path $p \in P$ is used and 0 otherwise. Using these variables, the VRPAR can be solved using the following set covering formulation:

$$\min \sum_{p \in P} (f + \Delta_p) z_p \tag{3}$$

$$\text{s.t.} \sum_{p \in P} s_p^j z_p \geq 1 \qquad \forall j \in J, \tag{4}$$

$$z_p \in \{0, 1\} \qquad \forall p \in P. \tag{5}$$

4. Branch and price algorithm on the road network

As the formulation (3) - (5) contains an exponential number of variables, we devise a branch and price algorithm to solve it. We explain the different components of this algorithm below.

4.1. Pre-processing of the road network

We first check the road network to see whether some restricted arcs can be converted into unrestricted arcs without changing the solution space. If the earliest arrival time at node $i \in N \setminus \{0\}$ from the depot is not less than l, then a vehicle can traverse all outgoing arcs of this node without waiting. Hence, if such nodes have restricted outgoing arcs, we drop the restrictions on these arcs.

Next, we search the road network to detect the arcs whose usage violates the workday duration limit. Let $\mathcal{E}_j$ be the earliest time a vehicle can arrive at node $j \in N \setminus \{0\}$ from the depot and $\mathcal{L}_j$ be the latest time a vehicle can depart from node j so that it can arrive at the depot by time T. We remove all arcs a such that $\mathcal{E}_{i_a} + \tau_a(\mathcal{E}_{i_a}) > \mathcal{L}_{j_a}$.

4.2. Initial solution

To produce a starting solution to the initial restricted master problem at the root node, we first solve all pairs shortest path problem in terms of travel costs on the underlying road network and construct a min-cost graph that only contains the arcs with the minimum distance between each pair of nodes. We use a greedy algorithm with two stages: clustering and routing. In the routing phase, the algorithm prioritises the arcs in the min-cost graph; it first searches the arcs in the min-cost graph to decide the next customer of a tour and uses the road network only when needed for workday duration feasibility.

First, we find a capacitated spanning tree where the capacity is defined as the vehicle capacity to cluster the customers. To do so, we use the min-cost graph, where the costs of the time-dependent arcs are multiplied by a positive parameter to prioritize the use of time-independent arcs. Then, we move to the routing part to create a tour (or tours) from each partition in this spanning tree. For each partition, we start the tour from the depot and at each iteration, we add the closest unvisited customer in the partition whose insertion does not violate the workday duration (also considering the time required to go back to the depot from this customer) to the tour. If no such

customer is found, we complete the current tour and start a new one. If no feasible insertion can be found at the beginning of a tour, it indicates that the set of customers in this partition cannot be visited using the arcs in the min-cost graph. In this case, we form the tour using the same criterion (the closest in time), but we use the road network instead of the min-cost graph. We apply this procedure until all customers are visited. Since the fleet size is unlimited, this algorithm always generates a feasible solution if one exists.

4.3. Pricing problem

This section is based on section 4.2. of Şahin and Yaman (2022) with the modification of allowing nodes to be visited more than once. For the sake of completeness, we present the algorithm in detail.

When we solve the LP relaxation of the above formulation using column generation, the pricing problem checks whether there exists a path with a negative reduced cost. We introduce graph $G = (N, A \cup A_s)$, where set A_s contains the copies of the incoming arcs of each customer j with $\alpha_j > 0$. Since the customers can be visited more than once on a road network, and the reduced cost of an arc depends on whether its head node is served, we use different arc sets for these two cases: if a customer is visited using an arc in A, the vehicle passes through this customer without serving it; if an arc in A_s is used, the customer must be served. Note that the arcs in set A can be used more than once in a feasible path, but not the ones in set A_s ($n_{ap} \in \{0,1\}$ for all $a \in A_s, p \in P$).

If we associate dual variable vector α to the set of the constraints (4), the pricing problem seeks to find a path $p \in P$ such that

$$\sum_{a \in A_p} n_{ap} w_a < -f$$

where $w_a = c_a$, $\forall a \in A$ and $w_a = c_a - \alpha_{j_a}$, $\forall a \in A_s$.

When $\alpha_j = 0$ for $j \in J$, the reduced costs of the incoming arcs of customer j in sets A and in A_s are equal. Meaning that any label serving customer j has the same cost as the label representing the same path, but that does not serve customer j. Since the former consumes the capacity resource more, it is always dominated by the latter, as explained in detail in the next section. To avoid generating these non-promising labels, A_s contains the incoming arcs of each customer j only when $\alpha_j > 0$.

We call a path elementary if every customer of the path is served at most once. Using this definition, the pricing problem can be defined as an ESPPRC with access restrictions where w_a is the cost of arc $a \in A \cup A_s$.

We make use of Boland, Dethridge, and Dumitrescu (2006)'s state-space augmenting algorithm to solve the ESPPRC. This algorithm relaxes elementarity constraints and imposes them iteratively as they are violated. We solve the pricing problem to optimality and add all the columns with a

negative reduced cost found during the process to the restricted master problem. The labelling algorithm can be executed by extending labels forward from origin to destination and backwards from destination to origin. Next, we explain both approaches, adapting them to our case.

4.3.1. Forward labelling Each forward label represents a path from the depot to an end node and its associated cost and resource consumption. When a forward label with end node i is extended using outgoing arc a of node i, arc a is added to the path from the depot to node i.

To determine the sequence for extending labels, we sort the labels based on specific criteria, prioritizing duration, followed by cost and capacity consumption to resolve ties. We subject the labels associated with the same node to a dominance test and discard the dominated labels. A label is considered dominated if all feasible extensions of the dominated label are also feasible for another label that is not more costly. The labelling algorithm terminates when all non-dominated labels have been extended.

To be able to detect a larger number of dominated labels, we make use of the notion of "unreachable nodes " as proposed by Feillet et al. (2004). We check the resource consumptions for the vehicle capacity and workday duration to identify unreachable nodes and mark such nodes as served. For the workday duration constraint, we use the latest arrival times calculated in the pre-processing phase. For each node $i \in N$, we compute the latest arrival time to node i to reach customer $j \in J \setminus \{i\}$ at time $\mathcal{L}_j$. If the time of a label associated with node i is greater than the computed time, then customer j is unreachable to this label. Since this procedure uses the fastest path for a given arrival time and the travel time function satisfies the FIFO property, times associated with these paths satisfy the triangle inequality by definition, which is required to make sure that the marked nodes are indeed unreachable.

Let f be a forward path starting at the depot and ending at node i_f at time $\tilde{t}_f \leq T$ such that access time windows and capacity constraints are respected and $\mathcal{S}$ be the set of *critical nodes* that are not allowed to be served more than once. We represent every forward path f with a label $L_f = (\tilde{c}_f, \tilde{q}_f, i_f, \tilde{t}_f, S_f)$ where $\tilde{c}_f$ is the reduced cost, $\tilde{q}_f$ is the total demand of the served customers and S_f is a binary *node-serve* resource vector of size $|\mathcal{S}|$ with $S_{\bar{r}}^i = 1$ if the i^{th} node in $\mathcal{S}$ has already been served or identified as *unreachable*, 0 otherwise for $i \in \{1, \dots |\mathcal{S}|\}$. To extend *node-serve* resources, we define a vector b_j of size $|\mathcal{S}|$ for each node $j \in J$: $b_j^i = 1$ if node j is the i^{th} node in $\mathcal{S}$, 0 otherwise.

We initialize the dynamic programming algorithm with a label for the depot defined as $\tilde{c}_f = 0$, $\tilde{q}_f = 0$, $i_f = 0$, $\tilde{t}_f = 0$ and $S_f = \vec{0}$. In the following, we explain the extension procedure in detail.

- Let $L_{f'}$ represent the label generated by extending L_f with $i_f = i \in N$ towards node $j \in N$ such that arc $a = (i, j) \in A$, $L_{f'}$ is constructed as follows:

- $\tilde{c}_{f'} = \tilde{c}_f + w_a$;
- $\tilde{q}_{f'} = \tilde{q}_f$;
- $i_{f'} = j$;
- $\tilde{t}_{f'} = \tilde{t}_f + \tau_a(\tilde{t}_f)$;
- $S_{f'} = S_f$ and, if $j \neq 0$, update the reachability of other critical nodes.

- If $i \in N$, $j \in J$ and $S_f^\top b_j = 0$, $L_{f'}$ is constructed for arc $a = (i,j) \in A_s$ (customer j is served) as follows:
  - $\tilde{c}_{f'} = \tilde{c}_f + w_a$;
  - $\tilde{q}_{f'} = \tilde{q}_f + q_j$;
  - $i_{f'} = j$;
  - $\tilde{t}_{f'} = \tilde{t}_f + \tau_a(\tilde{t}_f)$;
  - $S_{f'} = S_f + b_j$ and update the reachability of other critical nodes.

Due to the workday duration limit and vehicle capacity constraints, a forward label L_f is feasible only if $\tilde{t}_f \leq T$ and $\tilde{q}_f \leq Q$.

Let $L_f = (\tilde{c}_f, \tilde{q}_f, i_f, \tilde{t}_f, S_f)$ and $L_{f'} = (\tilde{c}_{f'}, \tilde{q}_{f'}, i_{f'}, \tilde{t}_{f'}, S_{f'})$ be two labels. To reduce the number of labels generated, we apply the following dominance rule: Label L_f dominates $L_{f'}$ if the following conditions are satisfied:

i) $i_f = i_{f'}$, ii) $\tilde{q}_f \leq \tilde{q}_{f'}$, iii) $\tilde{t}_f \leq \tilde{t}_{f'}$, iv) $S_f \leq S_{f'}$, v) $\tilde{c}_f \leq \tilde{c}_{f'}$

The first four conditions guarantee that each feasible extension of $L_{f'}$ is also feasible for L_f without incurring a higher cost, as indicated in the last condition. When all conditions are met with equality, we select one of the labels arbitrarily.

4.3.2. Backward labelling The labelling algorithm can also be executed by extending the labels from destination to origin. In the backward extension method, each label represents a path from its end node to the depot. When a backward label associated with node i is extended using incoming arc a of node i, arc a is added to the path from node i to the depot.

For backward labels, we initialize the algorithm with a backward label for the depot defined as $\tilde{c}_b = 0, \tilde{q}_b = 0, i_b = 0, \tilde{t}_b = T$ and $S_b = \vec{0}$. We apply the same extension method to the backward labels as in the forward labels but with a modification in the time update. To update the time of backward labels, we define the travel time of arc a where d_b is the time that the vehicle is on node j_a as $\hat{\tau}_a(d_b)$ and calculate it as follows:

$$\hat{\tau}_a(d_b) = \begin{cases} t_a & \text{if } a \notin A_R \vee (d_b - t_a) < e \vee (d_b - t_a) \geq l), \\ d_b - e + 1 & \text{otherwise} \end{cases} \tag{6}$$

and calculate the time of label $L_{b'}$ generated by extending L_b using arc $a \in A \cup A_s$ as $\tilde{t}_{b'} = \tilde{t}_b - \hat{\tau}_a(\tilde{t}_b)$. A backward label L_b is feasible only if $\tilde{t}_b \geq 0$ and $\tilde{q}_b \leq Q$.

We apply the following dominance rule: L_b dominates $L_{b'}$ if $i_b = i_{b'}$, $\tilde{q}_b \leq \tilde{q}_{b'}$, $\tilde{t}_b \geq \tilde{t}_{b'}$, $S_b \leq S_{b'}$ and $\tilde{c}_b \leq \tilde{c}_{b'}$.

4.3.3. Completion bounds We incorporate the completion bounds in the labelling algorithm, as described by Baldacci, Mingozzi, and Roberti (2011), to eliminate the labels that cannot produce columns with a negative cost earlier in the labelling process. A completion bound represents a lower bound on the cost required to complete a partial path. Using these bounds, we can show that some partial paths cannot be converted into negative-cost paths from origin to destination. Consequently, such paths can be eliminated.

As the relaxed problem gets tighter at each iteration in the state-space augmenting algorithm, the optimal value obtained at an iteration provides a lower bound for the next one. However, a path and its reverse might have different travel times in the VRP with access time restrictions because the travel time of an arc depends on the dispatch time from its tail. This means that a feasible path may become infeasible when reversed, and, as a result, the paths generated at the previous iteration cannot be directly used to complete the ones at the next iteration. To remedy this problem, we switch between forward and backward extensions in consecutive iterations. By switching the direction of the extension, the completion bounds at a given iteration can be obtained from the labels generated at the previous iteration. A full explanation of how the completion bounds are calculated can be found in Şahin and Yaman (2022).

4.3.4. Post-processing As the road network approach allows to pass through customers without serving, there might be unserved customers with a demand smaller than the remaining vehicle capacity in the generated paths. Since the master problem is formulated as a set covering problem, serving some unserved customers as well cannot weaken the corresponding cut added to the dual space. Therefore, when an elementary path with a negative cost is found, we start from the beginning of the path, and for each unserved customer, we check whether this customer can be served without violating the capacity constraints. If so, we modify the column to serve this customer. Otherwise, we move to the next unserved customer until all unserved customers are checked or the remaining vehicle capacity is smaller than the minimum demand of the remaining unserved customers.

4.3.5. Heuristic pricing As solving the pricing problem optimally can require a significant amount of time, we first solve the pricing problem on the min-cost graph to detect negative reduced cost columns and employ the exact method only if the heuristic algorithm fails.

The heuristic algorithm can be sped up by storing a limited number of labels at each node. When the label limit is reached, and a newly generated label has a lower ratio of cost to used capacity than any of the existing ones, the label with the highest ratio is removed, and the new label is stored in its place. While this approach may increase the number of pricing iterations, it typically yields a shorter overall computation time.

4.4. Branching scheme

If the optimal solution of the master problem is fractional, we apply hierarchical branching in which we first branch on the number of vehicles used and then branch on the arc flow variables.

Let v denote the number of vehicles used, that is, $v = \sum_{p \in P} z_p$. If this number is fractional, we add $\sum_{p \in P} z_p \leq \lfloor v \rfloor$ and $\sum_{p \in P} z_p \geq \lceil v \rceil$ to the master problem in the first and second child nodes, respectively. In the pricing problem, we incorporate the corresponding dual values to the cost of initial labels.

If the number of vehicles used is integer, we check whether there exists an arc a with fractional x_a value defined as $\sum_{p \in P: a \in A_p} z_p$. As the arcs can be used more than once in the road network, we add $\sum_{p \in P: a \in A_p} z_p \leq \lfloor x_a \rfloor$ and $\sum_{p \in P: a \in A_p} z_p \geq \lceil x_a \rceil$ to the master problem in the first and second child nodes, respectively. We also modify the reduced cost of arc a considering the corresponding dual values. When $\lfloor x_a \rfloor = 0$, modifying the master problem can be avoided. In this case, we do not inherit the columns that use arc a from the parent node and eliminate arc a from the graph to ensure that no new column that uses this arc is generated.

We use semi-strong branching to select an arc. First, we select a subset of candidate arcs $A_C \subset A$ with the most fractional flows. Then, for each candidate arc, we solve the master problems in both child nodes that would be obtained if this arc was selected for branching only with the columns generated up to this point. Then, we choose the arc with the highest estimated lower bound, that is $\min\{lb_a^1, lb_a^2\}$, where lb_a^1 and lb_a^2 are the lower bounds that would be obtained in the child nodes. We break the ties by the lower bound of the other child node.

In our preliminary experiments, we observed that at some nodes where arc a with $\lceil x_a \rceil > 1$ is used for branching, an optimal LP solution contains routes with cycles where no customer is served in between to satisfy the branching decision. To avoid such solutions, we prioritize the arcs with fractional values smaller than one in the selection of candidate arcs.

As shown in Ben Ticha et al. (2019), having no arc with a fractional flow does not suffice for integrality when a road network is used. Therefore if all arc flow variables are integer, they propose a branching scheme that enumerates all feasible routes in the support graph of the current solution in one branch and adds a constraint to the master problem in the second branch to impose the use of at least one arc that is not used in the current solution. In our implementation, when we encounter a node with a fractional solution supported by an integer flow, we do not branch on this node. If later this node is pruned by bound, we report the optimal solution. Otherwise, we report the best-found solution.

5. Solving VRPAR on the multi-graph

Another approach to solve routing problems with Pareto-optimal paths is constructing a multi-graph that contains an arc corresponding to each Pareto-optimal path. In this section, we explain

how to construct the multi-graph and what modifications are needed in the pricing problem to solve the VRPAR on a multi-graph.

5.1. Construction of the multi-graph

Before explaining the construction of the multi-graph, we will first define the terms used throughout this section. Since each arc in the multi-graph corresponds to a path in the underlying road network, we refer to the arcs in the multi-graph as "paths" from here on. Each path $p = (a_0, \ldots, a_k)$ with $i_{a_0} = i$ and $j_{a_k} = j$ is associated with a travel cost c^p and an arrival time function $\delta^p : [\mathcal{E}_i, \mathcal{L}_i] \to \mathbb{N}$ that returns the arrival time at node j for dispatch time d at node i. Note that $\delta^p(d)$ is the same as $\delta^p_{a_k}(d)$ defined in Section 2 and can be calculated using Equation 2.

Let p_1 and p_2 be two paths from node i to node j. Path p_2 *dominates* path p_1 for dispatch time d if $\delta^{p_2}(d) \leq \delta^{p_1}(d)$ and $c^{p_2} < c^{p_1}$ or $\delta^{p_2}(d) < \delta^{p_2}(d)$ and $c^{p_2} \leq c^{p_1}$. Path p_1 from node i to node j is *Pareto-optimal* for dispatch time d if there exists no path p_2 from i to j that dominates p_1.

We call the paths with time-dependent travel times *time-dependent paths*. Note that containing a restricted arc does not necessarily make a path time-dependent. For example, consider path ((1,8),(8,9),(9,2)) in Figure 1. Even though arc (9,2) is restricted, the earliest arrival time at node 9 using this path is 100, at which all arcs are accessible. In other words, the vehicle never waits at arc (9,2) regardless of the travel start time. Hence, the path is time-independent, as seen from its arrival time graph in the same figure. Using this observation, we call the restricted arcs that can be reached from the source node of the path before the end of the restricted period *critical restricted arcs* and define *time-dependent* paths as paths containing at least one critical restricted arc. The same arc could be critical for one path but not for another. As an example, in Figure 1, whereas arc (6,2) can be reached before the restricted period ends on path ((1,3),(3,4),(4,5),(5,6),(6,2)), meaning that it is critical for this path, it can be reached only after the restricted period ends on path ((1,8),(8,6),(6,2)). As all restricted arcs are closed to the traffic within the same period, a vehicle can wait at most at one critical arc on each path. For a given dispatch time, we call the critical arc at which the vehicle waits the *determining* arc.

As aforementioned, Pareto-optimal paths between a given node pair depend on the travel start time when the access restrictions are considered. However, the set of Pareto-optimal paths always contains a min-cost path for each node pair and start time because there is no other path with a smaller cost between the corresponding node pairs. Therefore, we first construct the min-cost graph by solving all pairs shortest path problem in terms of travel costs on the underlying network. To do so, we use a labelling algorithm where each label stores the travel cost, and a label is dominated only if there exists a cheaper label associated with the same node. This labelling algorithm generates all min-cost paths between each connected node pair.

While finding the min-cost paths, we also check whether the travel time of the path is time-dependent. Checking the time dependency of the min-cost paths is of great importance to the efficacy of the graph construction algorithm because if a min-cost path is time-independent, it implies that this path also has the shortest travel time for each start time (since the traversal times and costs are equal). Therefore, it dominates all other paths between the corresponding node pair for each start time, eliminating the need to check for other Pareto-optimal paths between this node pair.

If multiple min-cost paths exist between the same pair of nodes, we always choose a time-independent one if one exists. In the case of multiple time-independent min-cost paths, we arbitrarily choose one.

If a time-dependent min-cost path connects a node pair, we use the following proposition to find the breakpoint set of the arrival time function of this path.

PROPOSITION 1. *For a time-dependent path $p = (a_0, \ldots, a_k)$, the arrival time function δ^p is piecewise linear, and its breakpoint set is*

$$B^p = B_1^p \cup B_2^p,$$

where

$$B_1^p = \Big\{ \big(e - \sum_{j=0}^{i-1} t_{a_j}\big)_+ : i \in \{0, \ldots, k\} \text{ and } a_i \in A_R \Big\} \quad \text{and}$$

$$B_2^p = \Big\{ l - \sum_{j=0}^{i-1} t_{a_j} : i \in \{0, \ldots, k\}, a_i \in A_R \text{ and } l - \sum_{j=\underline{i}}^{i-1} t_{a_j} < e \ \forall \underline{i} \in [0, i-1], \quad a_{\underline{i}} \in A_R \Big\} \cup \big(\{0\} \setminus B_1^p\big).$$

Proof. The waiting times due to the access restrictions cause the arrival time function δ^p to have breakpoints. Waiting occurs at a restricted arc if a vehicle arrives at the start node of the arc in the interval $[e, l)$. A vehicle can arrive at the start node of arc $a_i \in A_R$ at time e, if it does not wait on any of the preceding arcs on the same path. The dispatch time from source node i_{a_0} to reach arc a_i at time e is equal to $e - \sum_{j=0}^{i-1} t_{a_j}$. If a restricted arc a_i cannot be reached before time e, $\sum_{j=0}^{i-1} t_{a_j} \geq e$, it indicates that the vehicle has to wait on this path even if it leaves source node i_{a_0} at time 0. The set B_1^p contains these dispatch times.

Reaching the start node of the restricted arc $a_i \in A_R$ at time l without violating the access restrictions requires arriving at each preceding restricted arc on path p, if any exists, before time e. Since no waiting time occurs until arc a_i, the dispatch time from source node i_{a_0} to reach arc a_i at time l is equal to $l - \sum_{j=0}^{i-1} t_{a_j}$. In addition, when the vehicle leaves node i_{a_0} at time $l - \sum_{j=0}^{i-1} t_{a_j}$, it reaches arc $a_{\underline{i}}$ where $\underline{i} \in [0, i-1]$ at time $l - \sum_{j=0}^{i-1} t_{a_j} + \sum_{j=0}^{\underline{i}-1} t_{a_j} = l - \sum_{j=\underline{i}}^{i-1} t_{a_j}$. Thus, for a restricted arc a_i in path p, if $l - \sum_{j=\underline{i}}^{i-1} t_{a_j} < e$ for all $\underline{i} \in [0, i-1]$ with $a_{\underline{i}} \in A_R$, $l - \sum_{j=0}^{i-1} t_{a_j}$ is one of the breakpoints of the arrival time function. □

PROPOSITION 2. *For a time-dependent path* $p=(a_0,\ldots,a_k)$, $B_1^p \cap B_2^p = \emptyset$

Proof. Let $i \in [0,k]$. For arc a_i, we have $l - \sum_{j=0}^{i-1} t_{a_j} \neq e - \sum_{j=0}^{i-1} t_{a_j}$ since $l \neq e$. For a successor arc $a_{\bar{i}}$ where $\bar{i} \in \{i+1,\ldots,k\}$, $\sum_{j=0}^{i-1} t_{a_j} < \sum_{j=0}^{\bar{i}-1} t_{a_j}$ Together with $l > e$, this implies that $l - \sum_{j=0}^{i-1} t_{a_j} > e - \sum_{j=0}^{\bar{i}-1} t_{a_j}$. If $l - \sum_{j=0}^{i-1} t_{a_j} \in B_2^p$, then $l - \sum_{j=0}^{i-1} t_{a_j} + \sum_{j=0}^{\underline{i}-1} t_{a_j} < e$ for a preceding restricted arc $a_{\underline{i}}$ by definition of B_2^p. □

To characterize the arrival time function, we first add T into set B_2^p for each $p \in A$. Then, we sort B^p in ascending order. We denote the i^{th} element of B^p by $B^p(i)$ and the order of determining arc for dispatch time $B^p(i) \in B_1^p$ on path p by j^i.

COROLLARY 1. *For a time-dependent path* $p=(a_0,\ldots,a_k)$, $\delta^p(d)$ *can be defined as*

$$\delta^p(d) = \begin{cases} l + \sum_{l=j^i}^{k} t_{a_l}, & \text{if } B^p(i) \leq d < B^p(i+1) \text{ and } B^p(i) \in B_1^p, \ \forall i \in \{0,\ldots,|B^p|-1\} \\ d + \sum_{l=0}^{k} t_{a_l}, & \text{if } B^p(i) \leq d < B^p(i+1) \text{ and } B^p(i) \in B_2^p, \ \forall i \in \{0,\ldots,|B^p|-1\} \end{cases}$$

To explain the breakpoints of an arrival time function on an example, consider path $p = ((1,3),(3,4),(4,5),(5,6),(6,2))$ in Figure 1. Since $a_3=(5,6)$ and $a_4=(6,2)$ cannot be reached before time 60, $\sum_{j=0}^{2} t_{a_j} = 60$ and $\sum_{j=0}^{3} t_{a_j} = 75$, $0 \in B_1^p$. On the other hand, $a_1=(3,4)$ satisfies the condition $\sum_{j=0}^{0} t_{a_j} = 2 < 60$: $B_1^p = \{0, 60-2\}$. For the breakpoints in B_2^p, as there is no preceding restricted arc before a_1, it satisfies the definition of B_2^p. Thus, B_2^p contains 100 - 2. Similarly, for the vehicle to reach a_3 at time 100, it needs to arrive at a_1 at time $100 - \sum_{j=1}^{2} t_{a_j} = 42$, which satisfies the definition of B_2^p. This means that B_2^p also contains $100 - \sum_{j=0}^{2} t_{a_j} = 40$. However, a_3 needs to be entered at time $100-15=85$ to reach a_4 at time 100, which does not comply with the access restrictions. Therefore, $B_2^p = \{40, 98\}$ and $B^p = \{0, 40, 58, 98, T\}$. The arrival time function corresponding to B^p by Corollary 1 is shown in Figure 1 by the pink line and can be defined as:

$$\delta^p(d) = \begin{cases} 100+15+5, & \text{if } 0 \leq d < 40 \\ d+80, & \text{if } 40 \leq d < 58 \\ 100+7+51+15+5, & \text{if } 58 \leq d < 98 \\ d+80, & \text{if } 98 \leq d < T \end{cases}$$

If there are multiple time-dependent min-cost paths between the same pair of nodes, we take the union of the breakpoints of these paths and choose the minimum arrival time at each time interval. We then merge the consecutive time intervals during which the arrival time function is continuous. We add one artificial path with the obtained arrival time function to the min-cost graph. We call the paths in the min-cost graph *merged min-cost paths* from here on. For instance, the arrival time

function of the merged min-cost path p corresponding to min-cost paths $((1,7),(7,10),(10,2))$ and $((1,3),(3,4),(4,5),(5,6),(6,2))$ in Figure 1 is as follows:

$$\delta^p(d) = \begin{cases} d+80, & \text{if } 0 \le d < 5 \\ 120, & \text{if } 5 \le d < 40 \\ d+80, & \text{if } 40 \le d < 58 \\ 165, & \text{if } 58 \le d < 85 \\ d+80, & \text{if } 85 \le d < T \end{cases}$$

After constructing the min-cost graph, we look for other Pareto optimal paths in the underlying network. As there is a limit on the workday duration, the Pareto-optimal paths between a given node pair and dispatch time can be found using the labelling algorithms designed for the SPPRC. We define the travel time as a resource and use the travel time function in Equation 1. We use the arrival time of the corresponding merged min-cost path at a given dispatch time as a time resource limit because it provides a tighter limit than the workday duration and enables us to eliminate more labels.

We denote the multi-graph by $G' = (N', A')$ where $N' = J \cup \{0\}$ and A' is the arc set containing Pareto-optimal paths. We denote the set of Pareto-optimal paths between nodes $i \in N'$ and $j \in N'$ by A'_{ij} and let $A' = \cup_{i,j \in N'} A'_{ij}$. We consider that each path $p = (a_0, \ldots, a_k) \in A'$ is characterized by its cost and travel time function and the set of Pareto-optimal paths contains distinct paths: if there are multiple Pareto-optimal paths between the same pair of nodes with the same cost and travel time for a given dispatch time, we abuse the definition and choose one of them arbitrarily and remove the others.

We use the following propositions derived from the piecewise linear and non-decreasing structure of the arrival time function to reduce the number of time periods at which Pareto-optimal paths must be searched.

PROPOSITION 3. *Let p_1 and p_2 be two paths between the same pair of nodes and $0 \le t_1 \le t_2 \le T$. Suppose that δ^{p_1} is constant within the interval $[t_1, t_2)$. If there exists $t' \in [t_1, t_2)$ for which p_1 is Pareto-optimal, then p_1 is Pareto-optimal for all dispatch times in $[t', t_2)$. Moreover, if there exists $t' \in [t_1, t_2)$ for which p_1 dominates p_2, then p_1 dominates p_2 for all dispatch times in $[t', t_2)$.*

Proof. Let p_1 and p_2 be paths between the same pair of nodes and $0 \le t_1 \le t_2 \le T$. Suppose that δ^{p_1} is constant within the interval $[t_1, t_2)$ and there exists $t' \in [t_1, t_2)$ for which p_1 is Pareto-optimal. Then, $\delta^{p_1}(t') \le \delta^{p_2}(t')$ for each path p_2 with $c^{p_2} = c^{p_1}$, and $\delta^{p_1}(t') < \delta^{p_2}(t')$ for each path p_2 with $c^{p_2} < c^{p_1}$. Let $t \in [t', t_2)$. Since $\delta^{p_1}(t) = \delta^{p_1}(t')$ and $\delta^{p_2}(t') \le \delta^{p_2}(t)$, it is easy to see that p_1 remains Pareto-optimal for dispatch time t as well. The same argument also applies to preserving the dominance as p_1 dominating p_2 at departure time $t' \in [t_1, t_2)$ means either $\delta^{p_1}(t') \le \delta^{p_2}(t')$ and $c^{p_1} < c^{p_2}$ or $\delta^{p_1}(t') < \delta^{p_2}(t')$ and $c^{p_1} \le c^{p_2}$. □

PROPOSITION 4. *Let p_1 and p_2 be two paths between the same pair of nodes and $0 \le t_1 \le t_2 \le T$. If δ^{p_1} is increasing within interval $[t_1, t_2)$ and if path p_1 dominates p_2 at time t_1, then p_1 dominates p_2 or $\delta^{p_1}(t) = \delta^{p_2}(t)$ and $c^{p_1} = c^{p_2}$ for all dispatch times t in $[t_1, t_2)$.*

Proof. Let p_1 and p_2 be paths between the same pair of nodes and $0 \le t_1 \le t_2 \le T$. Suppose that δ^{p_1} is increasing within the interval $[t_1, t_2)$ and p_1 dominates p_2 at time t_1. Let $t \in [t_1, t_2)$. Since δ^{p_1} is increasing in the interval $[t_1, t_2)$, we know that $\delta^{p_1}(t) = c^{p_1} + t$. Also we know that $\delta^{p_2}(t) \ge c^{p_2} + t$. Since p_1 dominates p_2 at time t_1, either $\delta^{p_1}(t_1) \le \delta^{p_2}(t_1)$ and $c^{p_1} < c^{p_2}$ or $\delta^{p_1}(t_1) < \delta^{p_2}(t_1)$ and $c^{p_1} \le c^{p_2}$. In the first case, $\delta^{p_1}(t) = c^{p_1} + t < c^{p_2} + t \le \delta^{p_2}(t)$ and in the second case $\delta^{p_1}(t) = c^{p_1} + t \le c^{p_2} + t \le \delta^{p_2}(t)$. So p_1 dominates p_2 or $\delta^{p_1}(t) = \delta^{p_2}(t)$ and $c^{p_1} = c^{p_2}$ for dispatch time t. □

We can also deduce from Proposition 4 that if the arrival time function of a merged min-cost path is increasing during a given time interval, as there is no path with a smaller cost, the merged min-cost path dominates all other paths. Consequently, there is no need to search for the Pareto-optimal paths for such time intervals, and the search can be started from the first breaking point where the arrival time function of the merged min-cost path becomes constant. By Proposition 3, if the merged min-cost path dominates all other paths at any point of a time interval where its arrival time is constant, the search for the Pareto-optimal paths can be continued from the next breakpoint.

If there are Pareto-optimal paths at any point of the time interval and if the arrival times of all Pareto-optimal paths are constant within the following interval of their arrival time functions, then the search can be continued from the minimum of the subsequent breakpoints of the arrival time functions of all Pareto-optimal paths by Proposition 3. On the other hand, if a Pareto-optimal path with an increasing arrival time function exists, it needs to be checked if this path gets dominated within its following interval. If it does not, the same procedure as in the constant arrival times, choosing the minimum of the succeeding breakpoints and continuing the search from there, can be applied by Proposition 4. If it does, this path is not considered to decide the next time period at which Pareto-optimal paths must be searched since the path dominating it also dominates the paths dominated by it. The outline of this procedure is given in Algorithm 1. This procedure generates a set of distinct Pareto-optimal paths.

After constructing the multi-graph, we search the graph to detect the arcs that cannot be used in a feasible solution. We remove all paths $p = (a_0, \ldots, a_k)$ with $i_{a_0} = i$ and $j_{a_k} = j$ such that $\delta^p(\mathcal{E}_i) > \mathcal{L}_j$.

Algorithm 1 The construction of the multi-graph

for each node $i_1 \in N'$ and $i_2 \in N'$ such that $q_{i_1} + q_{i_2} \leq Q$
  find all shortest paths between this pair
  for each shortest path $p = (a_0, a_1, a_2, \ldots, a_k)$ between i_1 and i_2
    if $\exists\, u \in \{0, \ldots, k\}$ such that $a_u \in A_R$ and $\sum_{j=0}^{u-1} t_{a_j} < l$ **then** mark p as time-dependent
    else mark p as time-independent and **break**
  if there exists a time-independent path **then** store this path as p^* and delete other min-cost paths
  else
    generate an artificial min-cost path p^* that has the earliest arrival time among all min-cost paths for each dispatch time and compute its breakpoints
    for $t_1 \in B_1^{p*}$
      let t_2 be the breakpoint of δ^{p^*} after t_1
      for $t_1' \in [t_1, t_2)$
        use the labelling algorithm for the SPRC, where time is used as a resource and bounded by $\delta^{p*}(t_1')$ and the vehicle leaves node i_1 at time t_1' to find all Pareto-optimal paths
        if no path is found **then break**
        else
          let P' be the set of Pareto-optimal paths between i_1 and i_2 for dispatch time t_1', P^c contain all paths $p \in P'$ such that the arrival function δ^p is constant in the interval containing t_1' and $P^i = P' \setminus P^c$
          $next \leftarrow t_2$
          for $p \in P'$
            let t_2' be the breakpoint of δ^p after t_1'
            if $p \in P^c$ **then** p is Pareto-optimal within $[t_1', t_2')$
              $next \leftarrow \min\{next, t_2'\}$
            else
              if $\exists t'' \in [t_1', t_2')$ such that $\min_{p' \in P^c \cup \{p*\}} \delta^{p'}(t_1') = \delta^p(t'')$ **then**
                p is Pareto-optimal within $[t_1', t'')$
              else p is Pareto-optimal within $[t_1', t_2')$
                $next \leftarrow \min\{next, t_2'\}$
          $t_1' \leftarrow next$

5.2. Branch and price algorithm on the multi-graph

In this section, we will explain the changes in the branch-and-price scheme when a multi-graph is used. As we refer to the arcs in the multi-graph as "paths", we use "workday" to denote the ordered list of paths that starts and ends at the depot. In addition to the feasibility conditions given in Section 3, we call a workday $w \in P$ feasible if it uses each path $p \in A_w$ exactly once, $n_{pw} = 1$, and if it visits each customer at most once. Each visited customer in workday w must be served. We define the set of all feasible workdays as $P' \subset P$. The master problem can be formulated the same as in formulation (3) - (5) where set P and index p are replaced by P' and w, respectively.

The pricing problem is an ESPPRC with access restrictions on multi-graph $G' = (N', A')$. It seeks to find a workday $w \in P'$ such that $\sum_{p \in A_w} w_p < 0$ where

$$w_p = \begin{cases} f + c^p - \alpha_{j_{a_k}}, \text{ if } i_{a_0} = 0 \\ c^p - \alpha_{j_{a_k}}, \quad \text{otherwise} \end{cases} \quad \forall p = (a_0, \ldots, a_k) \in A_w$$

To adapt the labelling algorithm to the multi-graph approach, we extend each label L_f with $i_f = i \in N$ towards node $j \in N$ through each path $p \in A'_{ij}$. We allow labels to be extended to a customer only if this customer receives service.

In the branching scheme, we observed in our preliminary experiments that using the standard branching rule, forbidding or enforcing the use of an arc, slows down the convergence of the algorithm. Therefore, we opt for enforcing or forbidding the successor of a node. If there exist two nodes $i \in N'$ and $j \in N'$ with fractional $\sum_{p \in A'_{ij}} x_p$ value, where $x_p = \sum_{w \in P': p \in A_w} z_w$, we branch in a way that node j must be visited right after node i in one branch, and node j cannot be visited right after node i in the other. In the first child node, if $i = 0$, we only inherit the columns that visit node j as the first customer or do not visit node j from the parent node. We also eliminate all incoming arcs of node j except for the arcs in A'_{ij} from the graph. Analogously, if $j = 0$, we only inherit the columns that visit node i as the last customer or do not visit node i from the parent node and eliminate all outgoing arcs of node i except for the arcs in A'_{ij} from the graph. If $i, j \in J$, we only inherit the columns that either visit nodes i and j consecutively or visit neither node i nor j. We also modify the graph by eliminating all outgoing arcs of node i and all incoming arcs of node j except for the arcs in A'_{ij}. In the second child node, we do not inherit the columns that visit nodes i and j consecutively and eliminate the arcs in A'_{ij} from the graph. We use semi-strong branching to select an arc to branch on, as explained in Section 4.4.

6. Computational experiments

In this section, we report the results of our computational experiments, where we investigate the computational effectiveness of our algorithm for both road network and multi-graph approaches. All experiments are carried out on a 64-bit machine with Intel Core i7 processor at 1.90 GHz and 16 GB of RAM using Java and CPLEX 12.10.

6.1. Computational performance for the road network and multi-graph approaches

In our experiments, we use 60 instances proposed by Ben Ticha et al. (2019), which were originally generated for the VRPTW on road network using the procedure in Letchford, Nasiri, and Oukil (2014). This procedure creates sparse graphs to resemble real-life road networks. The data set contains five instances for each instance group. We consider vehicles of capacity 150 and fixed cost of 35. To decide the workday duration, we first drop the access restrictions and compute

the shortest workday duration, call T', for which a two-way trip between the depot and each customer can be performed: $T' = \max_{j \in J}(\bar{c}_{0j} + \bar{c}_{j0})$ where $\bar{c}_{ij}$ is the length of a shortest path from i to j. Then, we define the restricted period under three different settings: close to the beginning $[0, \lceil 3T'/5 \rceil)$, close to the middle $[\lceil T'/5 \rceil, \lceil 4T'/5 \rceil)$ and close to the end of the workday $[\lceil 2T'/5 \rceil, T')$. After the restricted period is defined, for each customer j, we compute the earliest arrival time at this customer, $\mathcal{E}_j$, and the earliest time the vehicle can be back at the depot when it leaves customer j at time $\mathcal{E}_j$, call T'_j. These values can be computed by solving a shortest path problem in terms of travel times for given dispatch times. We then set T to $\lceil \max_{j \in J} T'_j * 1.1 \rceil$.

Table 2 The features of the road network and multi-graph for different instances

$[e,l)$			$[0, \lceil 3T'/5 \rceil)$			$[\lceil T'/5 \rceil, \lceil 4T'/5 \rceil)$			$[\lceil 2T'/5 \rceil, T')$		
inst.	$\lvert A \rvert$	$\lvert A_R \rvert$	$\lvert A^{mc} \rvert$	$\lvert A' \rvert$	G' gen. time	$\lvert A^{mc} \rvert$	$\lvert A' \rvert$	G' gen. time	$\lvert A^{mc} \rvert$	$\lvert A' \rvert$	G' gen. time
50-16	135	46	212	282	0.02	171	323	0.04	176	373	0.05
50-33	135	51	987	1293	0.06	826	1528	0.10	780	1868	0.18
100-25	278	91	560	717	0.02	381	828	0.03	407	1088	0.11
100-33	278	99	997	1184	0.07	723	1538	0.20	745	2191	0.20

To incorporate the access restrictions, we define a central area and define access restrictions for the arcs whose both endpoints are in this area. We put the grid's borders ten units away from the closest customer and define 5/8 of the grid in the center as the central area. We provide the average number of arcs in the road network, the average number of restricted arcs, the average number of paths in the multi-graph and in the min-cost graph after preprocessing is applied, and the average time spent to generate the multi-graph for each instance group in Table 2. These values show that the multi-graph has twice as many arcs as the min-cost graph on average. Moreover, thanks to the properties discussed in Section 5.1, generating a multi-graph takes 0.09 seconds on average for the tested instances.

First, we analyze the impact of the state space augmentation algorithm on the computation time for the road network and multi-graph approaches. We compare the straightforward labelling algorithm with the state-space augmentation algorithm. In both methods, first, the heuristic pricing algorithm is executed, and the exact algorithm is called only when the heuristic algorithm fails to find a negative cost workday. We set the time limit to three hours. We report the average values of the gap between the best obtained upper and lower bounds, the solution time, the number of nodes processed and the number of pricing iterations performed in Table 3 for the road network approach and in Table 4 for the multi-graph approach. In the gap column, we also report the number of instances that are solved to optimality within the time limit. When the column generation terminates at the root node, we solve the integer problem that contains all generated columns to

Table 3 Comparison of the straightforward and SSA labelling algorithms for the road network approach

			Straightforward					SSA				
$[e,l)$	$\|N\|$	$\|J\|$	gap(%) / solved	time	node	pricing iter	RootIP gap(%)	gap(%) / solved	time	node	pricing iter	RootIP gap(%)
$[0,\lceil 3T'/5\rceil)$	50	16	0 / 5	621.14	1924.60	11496.80	0	0 / 5	1583.79	2036.20	11463.40	0.98
	50	33	4.55 / 1	8671.91	139.60	928.60	4.76	3.27 / 1	8750.83	210.40	1342.20	4.35
	100	25	1.28 / 3	4537.22	1465.80	9344.80	2.37	1.08 / 3	4506.05	1098.20	6535.40	2.23
	100	33	2.66 / 2	9729.33	118.60	646.80	3.25	1.35 / 2	7529.73	331	1866.60	2.89
$[\lceil T'/5\rceil),\lceil 4T'/5\rceil)$	50	16	0 / 5	32.84	423.40	1626.60	1.24	0 / 5	64.05	387.80	1486.40	1.24
	50	33	1.90 / 3	5041.24	554.80	3438.80	2.37	1.96 / 3	5714.63	552	3549.60	2.55
	100	25	0.35 / 4	2482.94	1728.60	12094	1.37	0** / 5	952.20	373.80	2430	1.18
	100	33	1.09 / 3	4500.26	130.20	673.60	2.81	0.09 / 4	3763.16	346	1726.20	2.34
$[\lceil 2T'/5\rceil),\lceil T'\rceil)$	50	16	0 / 5	1.99	62.60	248	0.92	0 / 5	3.58	55.80	235.60	0.88
	50	33	0.62* / 2	6969.25	197.40	1273.60	1.69	1.13 / 3	6426.94	712.20	4232.60	1.86
	100	25	0.42 / 4	2248.97	162.80	1057.20	0.47	0.44 / 4	2268.19	114.60	774.80	0.51
	100	33	1.08 / 2	8691.60	550	3267.60	1.65	0.57 / 3	5810.15	560	3268.80	1.48

compute a better upper bound earlier in the tree. We report the average values of the gap between this upper bound and the best obtained lower bound in the RootIP gap column.

During our experiments, we observe one instance where the straightforward labelling algorithm reaches the time limit before solving the root node to optimality. As a valid lower bound is not obtained, we did not consider this instance in the average gap calculation. We use * in the gap column of the instance group it belongs to. We also observe one instance that is solved to optimality, but the optimality is not proved due to a fractional solution with an integer arc flow when the SSA algorithm is used on the road network approach. To denote this instance group, we use ** in the gap column.

Table 4 Comparison of the straightforward and SSA labelling algorithms for the multi-graph approach

			Straightforward					SSA				
$[e,l)$	$\|N\|$	$\|J\|$	gap(%) / solved	time	node	pricing iter	RootIP gap(%)	gap(%) / solved	time	node	pricing iter	RootIP gap(%)
$[0,\lceil 3T'/5\rceil)$	50	16	0 / 5	17.94	1557.40	6158.40	1.23	0 / 5	22.59	1719	6527.60	1.57
	50	33	1.57 / 2	7915.01	7393	34622.80	2.59	1.12 / 3	6385.13	15745.40	78170.80	2.28
	100	25	0 / 5	1057.22	23232.20	95108.80	2.44	0 / 5	854.40	22230.60	95359.60	1.07
	100	33	0.13 / 4	4796.71	48582.20	176805.60	0.80	0 / 5	3603.37	45367.40	161257	1.38
$[\lceil T'/5\rceil),\lceil 4T'/5\rceil)$	50	16	0 / 5	10.86	979.80	3134.40	2.27	0 / 5	10.17	687.40	2906	0.91
	50	33	1.50 / 2	6511.26	24579	115847.00	2.04	1.19 / 3	6326.36	30390.80	139695.80	2.05
	100	25	0 / 5	173.43	4547.40	19220.40	0.38	0 / 5	198.04	6907.40	24944.80	0.15
	100	33	0 / 5	1561.94	1452.20	7622.20	1.72	0 /5	628.69	1771	9635.80	1.86
$[\lceil 2T'/5\rceil),\lceil T'\rceil)$	50	16	0 / 5	1.84	80.20	292.60	0	0 / 5	0.72	33.40	128	0
	50	33	1.43 / 3	5717.43	3284.40	14935.20	2.04	0.99 / 3	4828.90	9043.20	43821.40	1.37
	100	25	0 / 5	207.72	1561	7805.40	0.29	0 / 5	130.53	1237.40	5968.80	0.44
	100	33	0.59 / 4	3076.41	1411.40	7149	1.09	0.68 / 4	2447.64	3230.60	18614.80	1.83

The results demonstrate that the SSA algorithm solves more instances to the optimality within the time limit, even though it increases the computation time for some instance groups in the road-network approach. Overall, 43 of 60 instances are solved to optimality in 66 minutes on average

with the SSA algorithm, whereas the straightforward labelling algorithm could solve 39 instances in 74 minutes on average. The SSA algorithm exhibits superior performance in the multi-graph approach as well. 53 of 60 instances are solved to optimality in 35 minutes on average when the SSA algorithm is used, and the number of solved instances reduces to 50 with an increased average computation time of 43 minutes when the straightforward algorithm is used.

As both the road network and the multi-graph approaches perform better when the SSA algorithm is used, we compare their performances when the SSA algorithm is used in Table 5. The results clearly demonstrate the superiority of the multi-graph approach over the road network approach for each instance group. The road network approach demonstrates a comparable performance with the multi-graph approach only for the instance group with 50 nodes and 33 customers where the restricted period is close to the middle of the workday.

We conjecture that this performance difference might result from the nice properties of the arrival time function. The multi-graph approach constructs a graph by leveraging these properties, as explained in Section 5.1, resulting in fewer non-promising partial routes generated in the pricing algorithm compared to the road network approach. Although using different branching schemes makes it difficult to compare the efficacy of both approaches to solve the pricing problems, the overwhelming superiority of the multi-graph approach suggests that the precalculations made in the graph construction phase significantly reduce the computational effort required to solve the pricing problem in the used instances.

Table 5 Comparison of the road network and the multi-graph approaches

$[e,l)$	$\|N\|$	$\|J\|$	Road network gap(%) / solved	time	node	pricing iter	RootIP gap(%)	Multi-graph gap(%) / solved	time	node	pricing iter	RootIP gap(%)
$[0, \lceil 3T'/5 \rceil)$	50	16	0 / 5	1583.79	2036.20	11463.40	0.98	0 / 5	22.59	1719	6527.60	1.57
	50	33	3.27 / 1	8750.83	210.40	1342.20	4.35	1.12 / 3	6385.13	15745.40	78170.80	2.28
	100	25	1.08 / 3	4506.05	1098.20	6535.40	2.23	0 / 5	854.40	22230.60	95359.60	1.07
	100	33	1.35 / 2	7529.73	331	1866.6	2.89	0 / 5	3603.37	45367.40	161257	1.38
$[\lceil T'/5 \rceil), \lceil 4T'/5 \rceil)$	50	16	0 / 5	64.05	387.80	1486.40	1.24	0 / 5	10.17	687.40	2906	0.91
	50	33	1.96 / 3	5714.63	552	3549.60	2.55	1.19 / 3	6326.36	30390.80	139695.80	2.05
	100	25	0** / 5	952.20	373.80	2430	1.18	0 / 5	198.04	6907.40	24944.80	0.15
	100	33	0.09 / 4	3763.16	346	1726.20	2.34	0 /5	628.69	1771	9635.80	1.86
$[\lceil 2T'/5 \rceil), \lceil T' \rceil)$	50	16	0 / 5	3.58	55.80	235.60	0.88	0 / 5	0.72	33.40	128	0
	50	33	1.13 / 3	6426.94	712.20	4232.60	1.86	0.99 / 3	4828.90	9043.20	43821.40	1.37
	100	25	0.44 / 4	2268.19	114.60	774.80	0.51	0 / 5	130.53	1237.40	5968.80	0.44
	100	33	0.57 / 3	5810.15	560	3268.80	1.48	0.68 / 4	2447.64	3230.60	18614.80	1.83

6.2. The impact of the access restrictions

In the presence of access restrictions, one of the key decisions that must be made is whether to wait on a closed street until it reopens or to take a more costly arc to save time. To assess the effect of access restrictions on optimal solutions, we look at the characteristics of the routes. We

consider only the instances solved to optimality. We report the percentage of waiting time to the total travel time, the number of vehicles used, and the number of routes in which a non-min-cost path is used in Table 6. If an optimal solution is not found within the time limit, we use dash for each column of this instance. To assess the impact of Pareto-optimal paths on the travel cost, we also compare the problems where all Pareto-optimal paths are considered and where only min-cost paths are considered. We report the savings in the cost-saving column. We use inf for the instances for which the problem is infeasible when only min-cost paths are considered.

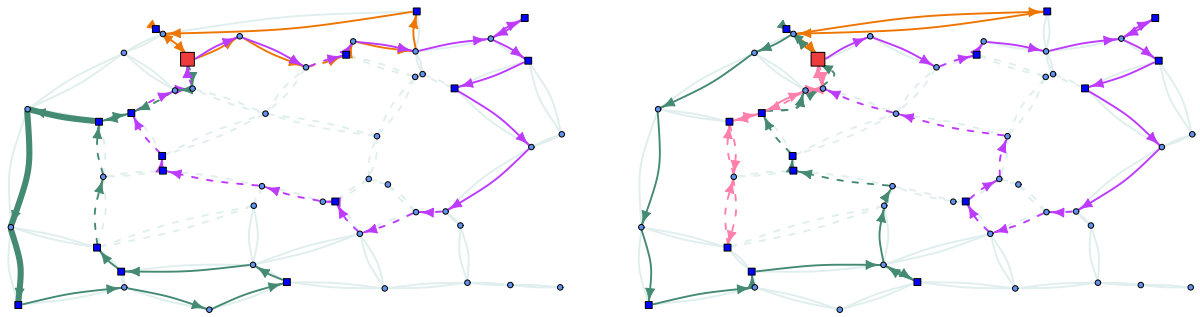
Table 6 The characteristics of optimal solutions

inst.	$[0, \lceil 3T'/5 \rceil)$				$[\lceil T'/5 \rceil, \lceil 4T'/5 \rceil)$				$[\lceil 2T'/5 \rceil, T')$			
	waiting time (%)	**# of routes**	**# of alt. routes**	**cost saving (%)**	**waiting time (%)**	**# of routes**	**# of alt. routes**	**cost saving (%)**	**waiting time (%)**	**# of routes**	**# of alt. routes**	**cost saving (%)**
50-16-1	0	3	2	inf	11.35	3	1	15.90	0	4	2	inf
50-16-2	15.26	3	1	inf	7.48	3	1	0.24	16.46	3	0	0
50-16-3	9.35	3	0	0	15.69	3	1	2.20	0	3	0	0
50-16-4	28.57	3	1	4.60	10.87	3	0	0	20.25	3	0	0
50-16-5	31.67	3	0	0	5.33	3	0	0	11.73	3	0	0
50-33-1	-	-	-	-	-	-	-	-	25.74	5	0	0
50-33-2	27.88	5	0	0	12.20	5	0	0	10.65	5	1	0.35
50-33-4	38.08	5	1	1.87	21	5	1	inf	-	-	-	-
50-33-5	25.59	5	1	0.91	16.67	5	0	0	18.11	5	0	0
100-25-1	18.89	4	0	0	20	4	0	0	30.13	4	0	0
100-25-2	36.41	4	0	0	8.88	4	2	8.25	9.56	4	1	1.32
100-25-3	12.14	4	1	1.98	20.79	4	0	0	10.50	4	0	0
100-25-4	10.27	4	1	3.61	8.28	4	1	1.70	16.32	4	0	0
100-25-5	40.32	4	0	0	17.43	5	0	0	18.29	4	1	1.62
100-33-1	24.20	5	0	0	19.09	5	0	0	-	-	-	-
100-33-2	36.76	5	0	0	5.77	5	1	1	15.76	5	0	0
100-33-3	27.72	5	1	1.24	27.19	5	0	0	33.53	5	0	0
100-33-4	42.41	5	0	0	21.77	5	2	2.13	23.87	5	0	0
100-33-5	38.78	6	0	0	9.57	5	0	0	16.65	5	0	0

The results demonstrate that choosing between waiting on a closed street or taking a more costly path is not straightforward. In most instances where a non-min-cost path is used, vehicles still choose to wait on some streets. Additionally, waiting times and the choice of using a non-min-cost path vary significantly even within the same instance groups, emphasizing the instance-specific nature of the decision. These findings show the need for optimization methods in addressing the trade-off between waiting times and travel costs. We also observe that considering only min-cost paths leads to infeasibility in four of 53 instances.

To demonstrate the importance of alternative paths in VRPAR, Figure 4 depicts the optimal solutions of instance 50-16-1 on the road network and on the min-cost graph, where the restricted period is defined as $[\lceil T'/5 \rceil, \lceil 4T'/5 \rceil)$. The optimal solution obtained on the min-cost graph is also converted into a road network solution to demonstrate the differences and similarities between the two solutions more clearly. The circles represent the road junctions, the blue rectangles represent the customers, and the red rectangle represents the depot. The dashed lines are used to represent the restricted arcs. Different colours are used for the workday of each vehicle. The optimal value

of the first problem is 1058, and it increases to 1258 when only min-cost paths are considered. In the road network solution, the vehicle whose workday is represented by a green line uses a non-min-cost path between two customers, which is represented by a thicker line. Visiting the same set of customers in the same order is infeasible on the min-cost graph because the vehicle has to wait on the min-cost path between these two customers, and this waiting time causes exceeding the workday duration. Not considering this non-min-cost path leads to using one more vehicle but also increases the total travel cost compared to that of the road network solution.



(a) road network **(b) min-cost graph**

Figure 4 An optimal solution of 50-16-1 instance for different graph types

7. Conclusion

In this study, we presented two branch and price algorithms for VRPAR, one based on the road network and the other one based on the multi-graph. The results of the conducted experiments suggest the superiority of the multi-graph approach over the road network approach for the tested instances. We believe this performance difference results from being able to incorporate the problem structure in the multi-graph approach but not in the road network approach. During our preliminary experiments, we observed that the road network approach struggles, while solving the pricing problems, with routes containing cycles where no customer is served. These cycles can have a negative cost due to the branching decisions, and it is not obvious to eliminate routes containing such cycles without jeopardizing optimality. We also observe that such routes are used in an optimal LP solution of some nodes to satisfy branching decisions, resulting in poorer LP bounds and degrading the algorithm's performance. Developing a technique to avoid these cycles in the road network approach is an important future research direction.

The implementation of access restrictions may cause long waiting times during which vehicles need to find a parking space. Starting a workday later, that is, waiting at the depot at the beginning of the workday, may reduce the waiting times, but it does not necessarily eliminate the need for waiting. For example, consider the workday shown in Figure 5, where node 0 represents the depot,

dashed lines represent the restricted arcs, and the vehicle must return to the depot by time 6. If the vehicle leaves the depot at time zero, it waits at node 2 for one time unit and is back at the depot at time 6. To avoid waiting at node 2, the vehicle should leave the depot the earliest at time one. However, starting the workday at time one causes infeasibility because now the vehicle needs to wait at node 1 until time 4. As this small example demonstrates, there may be some workdays with no workday start time that eliminates the waiting time. In this study, we assume the presence of a parking space at each node. Incorporating the availability of parking spaces is a promising extension of our current work.

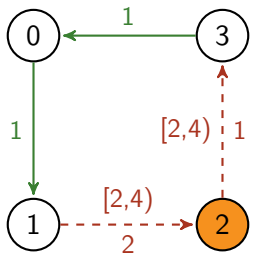


Figure 5 **Example demonstrating the need for waiting**

Another limitation of our study is allowing streets to be closed within a single time period. A more general setting is to consider several periods where streets are closed to traffic at different times throughout the day, such as the start and end of a school day. When access restrictions are defined only for one time period and are the same for all arcs, the cardinality of the breakpoint set of the arrival time function of a time-dependent path is bounded by twice the number of restricted arcs on this path. However, in the presence of multiple restricted periods, this set can grow quickly because the vehicles might wait on more than one restricted arc on the same path. Even though the resulting arrival time function is still piecewise linear, this increase in the number of breakpoints might significantly increase the size of the multi-graph and the required time to construct this graph. On the other hand, the road network approach can be used to solve the problem with multiple restricted periods with a slight modification in the arrival time function. Considering multiple restricted periods is an interesting extension of our current work.

Another promising direction of future work is to consider *low emission zones* and incorporate small vehicles into the fleet. In this extension, since the small vehicles have access to the restricted areas, the routes of small vehicles can be planned on a min-cost graph if the traversal time of each arc is equal to (or a positive multiple of) its travel cost. On the other hand, the routes of the large vehicles still need to be planned either on a road network or a multi-graph, utilizing the methodologies proposed in our current work.

Acknowledgments

We would like to thank Prof. Roel Leus for his valuable suggestions.

References

Baldacci R, Mingozzi A, Roberti R, 2011 *New route relaxation and pricing strategies for the vehicle routing problem. Operations Research* 59(5):1269–1283.

Bektaş T, Laporte G, 2011 *The pollution-routing problem. Transportation Research Part B: Methodological* 45(8):1232–1250.

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.

Ben Ticha H, Absi N, Feillet D, Quilliot A, Van Woensel T, 2021 *The time-dependent vehicle routing problem with time windows and road-network information. Operations Research Forum*, volume 2, 1–25 (Springer).

Boland N, Dethridge J, Dumitrescu I, 2006 *Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters* 34(1):58–68.

Corberán Á, Eglese R, Hasle G, Plana I, Sanchis JM, 2021 *Arc routing problems: A review of the past, present, and future. Networks* 77(1):88–115.

Dabia S, Ropke S, Van Woensel T, De Kok T, 2013 *Branch and price for the time-dependent vehicle routing problem with time windows. Transportation Science* 47(3):380–396.

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

Franceschetti A, Honhon D, Van Woensel T, Bektaş T, Laporte G, 2013 *The time-dependent pollution-routing problem. Transportation Research Part B: Methodological* 56:265–293.

Garaix T, Artigues C, Feillet D, Josselin D, 2010 *Vehicle routing problems with alternative paths: An application to on-demand transportation. European Journal of Operational Research* 204(1):62–75.

Grosso R, Muñuzuri J, Escudero-Santana A, Barbadilla-Martín E, 2018 *Mathematical formulation and comparison of solution approaches for the vehicle routing problem with access time windows. Complexity* 2018:1–10.

Huang Y, Zhao L, Van Woensel T, Gross JP, 2017 *Time-dependent vehicle routing problem with path flexibility. Transportation Research Part B: Methodological* 95:169–195.

Jaballah R, Veenstra M, Coelho LC, Renaud J, 2021 *The time-dependent shortest path and vehicle routing problem. INFOR: Information Systems and Operational Research* 1–31.

Letchford AN, Nasiri SD, Oukil A, 2014 *Pricing routines for vehicle routing with time windows on road networks. Computers & Operations Research* 51:331–337.

Muñuzuri J, Grosso R, Cortés P, Guadix J, 2013 *Estimating the extra costs imposed on delivery vehicles using access time windows in a city. Computers, Environment and Urban Systems* 41:262–275.

Şahin MK, Yaman H, 2022 *A branch and price algorithm for the heterogeneous fleet multi-depot multi-trip vehicle routing problem with time windows. Transportation Science* 56(6):1636–1657.

Setak M, Habibi M, Karimi H, Abedzadeh M, 2015 *A time-dependent vehicle routing problem in multigraph with fifo property. Journal of Manufacturing Systems* 35:37–45.

Ticha HB, Absi N, Feillet D, Quilliot A, 2017 *Empirical analysis for the vrptw with a multigraph representation for the road network. Computers & Operations Research* 88:103–116.