https://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–23
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

The Electric Vehicle Routing and Overnight Charging Scheduling Problem on a Multigraph

Daniel Yamín,[a] Guy Desaulniers,[b,*] Jorge E. Mendoza[c]

[a] Centro para la Optimización y Probabilidad Aplicada (COPA), Departamento de Ingeniería Industrial, Universidad de los Andes, Bogotá 111711, Colombia; [b] Polytechnique Montréal and GERAD, Montreal, Quebec H3C 3A7, Canada; [c] HEC Montréal, Montreal, Quebec H3T 2A7, Canada

*Corresponding author

Contact: d.yamin@uniandes.edu.co, https://orcid.org/0000-0001-8770-2379 (DY); guy.desaulniers@gerad.ca, https://orcid.org/0000-0003-4469-9813 (GD); jorge.mendoza@hec.ca, https://orcid.org/0000-0003-2473-2655 (JEM)

Received: November 10, 2023
Revised: April 14, 2024; July 9, 2024
Accepted: July 31, 2024
Published Online in Articles in Advance: September 2, 2024

https://doi.org/10.1287/ijoc.2023.0404

Abstract. In the electric vehicle (EV) routing and overnight charging scheduling problem, a fleet of EVs must serve the demand of a set of customers with time windows. The problem consists in finding a set of minimum cost routes and determining an overnight EV charging schedule that ensures the routes' feasibility. Because (i) travel time and energy consumption are conflicting resources, (ii) the overnight charging operations take considerable time, and (iii) the charging infrastructure at the depot is limited, we model the problem on a multigraph where each arc between two vertices represents a path with a different resource consumption trade-off. To solve the problem, we design a branch-price-and-cut algorithm that implements state-of-the-art techniques, including the *ng*-path relaxation, subset-row inequalities, and a specialized labeling algorithm. We report computational results showing that the method solves to optimality instances with up to 50 customers. We also present experiments evaluating the benefits of modeling the problem on a multigraph rather than on the more classical 1-graph representation.

History: Accepted by Andra Lodi, Area Editor for Design and Analysis of Algorithms—Discrete.
Funding: This work was supported by the Natural Sciences and Engineering Research Council of Canada through the Discovery grants [Grant RGPIN-2023-03791]. It was also partially funded by HEC Montréal through the research professorship on Clean Transportation Analytics.
Supplemental Material: The software that supports the findings of this study is available within the paper and its Supplemental Information (https://pubsonline.informs.org/doi/suppl/10.1287/ijoc.2023.0404) as well as from the IJOC GitHub software repository (https://github.com/INFORMSJoC/2023.0404). The complete IJOC Software and Data Repository is available at https://informsjoc.github.io/.

Keywords: electric vehicle routing • charging scheduling • multigraph • branch-price-and-cut • labeling algorithm • city logistics

1. Introduction

In recent years, logistics companies dealing with urban distribution have started transitioning from internal combustion engine vehicles (ICEVs) to electric vehicles (EVs). One of the main drivers of this transition is that EVs alleviate many of the unsustainable practices of present-day logistics. For instance, they produce minimal noise and no local greenhouse gas emissions, allowing companies to comply with noise restrictions and emission targets that are usually imposed in inner-city areas (Davis and Figliozzi 2013). Furthermore, EVs enable companies to improve their market position by promoting a "green" image, a competitive advantage due to the increasing environmental consciousness of consumers (Schneider et al. 2014). From a cost perspective, EVs are becoming more attractive as a result of recent technological developments, spiking oil prices, and government incentives (Desaulniers et al. 2016). Nonetheless, EVs still have significant disadvantages over ICEVs, such as a high acquisition cost, limited driving range, slow recharging times, and a lack of widespread recharging infrastructure (Pelletier et al. 2016, Bektaş et al. 2019). Although the acquisition cost is expected to decrease over the coming years through economies of scale and technology innovations (Florio et al. 2021), the restrictive driving range and recharging hassles must be addressed before EVs can be widely adopted in urban logistics.

To deal with the operational limitations of EVs, early research focused on electric vehicle routing problems (E-VRPs) with en-route charging policies (Kullman et al. 2021b). As the name suggests, these problems consist in designing a set of routes that may include out-of-depot charging operations. These operations may be carried on at public (i.e., open access) or private (i.e., access limited to the fleet owner) charging stations. En-route charging is convenient because it allows extending the range of the EVs without back-and-forth trips to the depot, a real issue because the latter is rarely located close to the customers. However, en-route charging also has disadvantages. First, access to out-of-depot charging infrastructure is not straightforward. Many companies cannot deploy private chargers because of their high installation and operating cost. On the other hand, relying on public chargers can be problematic because waiting times may be long and difficult to predict. Second, charging the EVs out of their depot is usually more expensive. This is true even at private chargers because the routes (and the charging operations) are typically executed during the day when time-dependent energy prices are at their peak (Pelletier et al. 2018). Last but not least, en-route charging may yield undesirable driver idle times and cargo security concerns.

With the constant improvement in EV driving ranges, en-route charging is becoming less of a necessity. Today, it is possible to charge the EVs only at the depot (overnight or between shifts) and still be able to design efficient urban routes (Morganti and Browne 2018). The latter gives birth to a new breed of coupled routing and charging scheduling problems in which the decision maker must simultaneously plan the routes and the EVs' charging operations. These problems are especially challenging to solve because the two components (routing and scheduling) are intimately—and sometimes conflictingly—intertwined. For instance, designing long routes that visit many customers may be ideal from a routing (cost) point of view. However, those routes will likely require long charging operations that may be impossible to schedule because of the number of chargers available at the depot. On the other hand, short routes leading to easier-to-schedule charging operations may be inefficient or lead to an (unnecessary) increase in the number of vehicles used. Similarly, good coordination between the routing and the scheduling components is essential to preserve the life of the EVs' batteries. As Pelletier et al. (2018) pointed out, to prevent calendar aging, a charging operation should finish as close as possible to the departure time of the associated vehicle. Needless to say, achieving this coordination leads to harder combined routing and charging scheduling problems.

In this paper, we introduce, to the best of our knowledge, the first variant in this problem family: the multigraph-based electric vehicle routing and overnight charging scheduling problem (mE-VRSPTW). Our problem embeds classical vehicle routing problem (VRP) features such as customer time windows and vehicle load capacity and state-of-the-art E-VRP features such as battery capacity, nonlinear charging functions, and limited charging infrastructure. To allow better coordination between the routing and scheduling components, we model the problem on a multigraph in which there may be more than one arc connecting each pair of customers (and the depot). These arcs allow us to model paths (on the road network) offering different trade-offs between resources, such as travel distance and energy consumption. These trade-offs are well documented in the EV literature—see, for example, Asamer et al. (2016), but are seldom exploited in E-VRP models and algorithms.

We devise a branch-price-and-cut (BPC) algorithm to solve this new and challenging problem. In our algorithm, the master problem handles customer covering and depot charging capacity, and the pricing problem builds routes and their associated charging schedules. The algorithm implements state-of-the-art techniques, including the *ng*-path relaxation, subset-row inequalities, and a specialized labeling algorithm. We also present problem-specific dominance and branching rules. We report computational experiments on instances with up to 50 customers adapted from VRP with time windows (VRPTW) instances. Our algorithm delivers optimal solutions for all of them in less than 53 minutes. We also present experiments evaluating the benefits of modeling the problem on a multigraph rather than on the more classical 1-graph representation. On the former, our algorithm finds better solutions than on the latter at the expense of higher computational times.

The remainder of this paper is organized as follows. Section 2 reviews related literature. Section 3 formally defines the mE-VRSPTW. Section 4 describes a BPC algorithm to solve this problem. Section 5 reports our computational results. Finally, Section 6 concludes the paper and outlines future research.

2. Related Literature

Our research sits at the crossroads of EV routing, on-site EV charging scheduling, and vehicle routing on road networks. In this section, we provide a concise overview of pertinent literature in these domains, specifically chosen for its direct relevance to our problem definition and solution method. As it stands, in the EV routing subsection, we focus mainly on exact methods, but we refer the reader to Schneider et al. (2015), Goeke and Schneider (2015), Hiermann et al. (2016), and Schiffer and Walther (2018) for research on heuristics and metaheuristics for E-VRPs and closely related problems.

2.1. Electric Vehicle Routing Problems

The first exact method for large-sized E-VRPs was introduced by Desaulniers et al. (2016). They studied four variants of the E-VRP with time windows, considering (i) a single or (ii) multiple recharging stops and (iii) full or

(iv) partial recharges. In all these variants, they assumed that the recharging time is a linear function of the amount of energy recharged. They presented a BPC algorithm in which the master problem is variant-agnostic and reinforced with valid inequalities. In contrast, the pricing problem is variant-specific and solved by a sophisticated labeling algorithm. Later, Desaulniers et al. (2020) improved this algorithm by fixing some variables associated with routes traversing a sequence of two arcs to zero.

Montoya et al. (2017) investigated the nonlinear nature of the EV battery charging process and found that this behavior can be accurately approximated by a piecewise-linear function. Based on this approximation, they developed a mixed-integer linear program (MILP) to solve a complex E-VRP with a nonlinear charging function, heterogeneous chargers (i.e., multiple charging technologies), multiple recharging stops, and partial recharges. Their model employs the charging station replication paradigm, in which the charging station vertices are replicated, and the number of stops at each of these vertices is limited to one. This procedure helps track the travel time, distance, and state of charge (SoC) of routes, but can potentially yield intractable models because of symmetry issues. Their computational experiments underscored the importance of considering the charging process nonlinearity, as neglecting it could lead to infeasible or excessively costly solutions.

In reexamining the work of Montoya et al. (2017), Froger et al. (2019) introduced two enhanced MILP formulations. The first of these formulations is rooted in the station replication paradigm, but deviates from that of Montoya et al. (2017) by using arc-indexed variables to track travel time and distance instead of vertex-indexed ones. Their second formulation, on the other hand, relies on the recharging-path paradigm, where a path represents a sequence of charging station stops between nodes. This approach demands the enumeration of the possible charging station visit sequences between every pair of vertices. They devised dominance rules to reduce the number of paths in the model. Notably, their computational experiments demonstrated that these new formulations deliver faster solution times compared with the one proposed by Montoya et al. (2017). This improvement is primarily attributed to a better linear relaxation bound.

In practical scenarios, charging stations often feature a fixed and limited number of chargers, a characteristic that has been repeatedly overlooked in the E-VRP literature. Recent studies have begun to bridge this gap by incorporating capacity constraints at charging stations. For instance, Bruglieri et al. (2021) studied the green VRP with multiple refueling stops, full refueling, and capacitated alternative fuel stations. They assume that when a vehicle visits a station, it refuels for a fixed duration, reaching its maximum capacity. They introduced a path-based MILP, which is solved by a cutting-plane algorithm that separates fueling station capacity constraints when necessary.

Froger et al. (2022) addressed the E-VRP with a piecewise-linear recharging function, heterogeneous chargers, multiple recharging stops, partial recharges, and capacitated charging stations. As in their prior work, they devised a path-based MILP that enforces charger capacity constraints through sequencing variables and is solved using a commercial solver. Their experiments revealed that this approach lacks efficiency, primarily because of the model's weak linear relaxation, and highlighted a critical finding: neglecting explicit modeling of charging station capacity can lead to infeasible solutions in practical applications.

Lam et al. (2022) investigated the E-VRP with time windows, a nonlinear recharging function, multiple recharging stops, partial recharges, and capacitated charging stations. They proposed a BPC algorithm in which column generation (CG) addresses the routing component whereas constraint programming handles the scheduling component. When the method identifies an integer routing solution, it checks for the existence of a corresponding feasible charging schedule. When such a schedule does not exist, they derive a logic-based Benders cut, leading to the elimination of the routing solution from the search space. To enhance the effectiveness of the combinatorial Benders cuts, they introduced a strategy based on conflict analysis.

2.2. Electric Vehicle On-Site Charging Scheduling

All studies reviewed above have focused on problems in which charging operations occur off-site, that is, outside the depot. However, as emphasized by Kullman et al. (2021a), many EV operators avoid this approach in order to circumvent the uncertainty of charging station availability and minimize driver idle time. Instead, they prefer to charge their vehicles on-site, that is, at the depot, before they leave to perform their routes. Consequently, on-site charging scheduling problems have received substantial attention in the literature.

In a pioneering work, Sassi and Oulamara (2014) investigated the challenge of allocating a mixed fleet of EVs and ICEVs to a set of predetermined routes with the dual objective of maximizing EV usage and minimizing charging costs. The EVs recharge on-site between two assigned routes, taking into account time-dependent energy costs and grid power capacity constraints. In a related study, Pelletier et al. (2018) tackled a more realistic on-site charging scheduling problem, extending the planning horizon to multiple days and incorporating nonlinear charging functions. They further refined the objective function to include considerations for battery degradation and facility-related demand costs. Both Sassi and Oulamara (2014) and Pelletier et al. (2018) modeled their respective problem as a MILP that can be solved by a commercial solver.

The previously discussed studies focused on logistics scenarios, but the challenges of on-site charging scheduling are equally prevalent in public transit. Alvo et al. (2021) conducted research on a bus dispatching problem with a mixed fleet of electric and diesel buses and a limited number of chargers. In this study, the task involves assigning buses to trip schedules and creating a plan for their battery charging, assuming a linear charging function. To solve the problem, they introduced a Benders-type decomposition approach, where a master problem is responsible for assigning buses to trip schedules, and a subproblem is tasked with producing a battery charge plan for the selected set of bus trip schedules. Zhou et al. (2022) also conducted a study on bus dispatch operations at a public transport terminal. They focused on solving the issue of determining trip assignments and charging schedules for electric buses operating on a specific bus line. In their approach, they took into account a nonlinear charging function as well as the costs associated with battery degradation. To tackle the problem, they designed two MILPs, which they enhanced by incorporating problem-specific valid inequalities. More recently, Vendé et al. (2023) solved a multiday electric bus assignment and overnight recharge scheduling problem using a MILP and a set of decomposition-based matheuristics. In their problem, they consider heterogeneous chargers, nonlinear charging functions, and calendar aging costs.

Note that, unlike in E-VRPs, in on-site charging scheduling problems, vehicles either are preassigned to routes or must be assigned to a predefined set of routes. In essence, the routing component is significantly reduced or virtually absent in this line of research. Furthermore, MILP formulations, when executed on commercial solvers, exhibit relatively good scalability for these problems. As a result, there is a notable scarcity of specialized exact methods in the existing literature (Perumal et al. 2022).

2.3. Vehicle Routing on Road Networks

VRPs, in particular E-VRPs, are traditionally defined using customer-based 1-graphs. In these graphs, vertices represent locations of interest, encompassing customers and depots, whereas arcs represent the optimal paths connecting these locations. These optimal paths are typically computed with a focus on a single criterion, such as minimizing travel cost, distance, or time. However, in practical scenarios, multiple alternative paths often exist when traveling between two locations, each offering different trade-offs between two or more criteria. VRPs taking this feature into account are commonly known as VRPs on road networks (Ben Ticha et al. 2018).

In addition to depot and customer locations, road networks contain nodes representing road junctions. Therefore, arcs in these networks represent street segments rather than complete paths connecting two locations. Needless to say, in road networks, there are typically multiple paths between two customers or a customer and a depot.

The literature reports two primary approaches for solving VRPs on road networks. The first utilizes the road network to construct a customer-based multigraph, including all nondominated paths between every pair of customer vertices and the depot. The second involves working directly with the road network graph, as described above.

In their study, Ben Ticha et al. (2017) investigated the feasibility and benefits of exploiting a multigraph for the VRPTW. To draw meaningful conclusions and provide a fair and comprehensive comparison with the classical 1-graph, they implemented a branch-and-price algorithm to solve the problem on the multigraph road network and conducted an empirical analysis using real-world instances. They compared the solutions obtained on the multigraph to those obtained on two customer-based 1-graphs: the min-cost graph (representing the cheapest path) and the min-time graph (representing the fastest path). Their findings revealed that although solving the problem on the multigraph representation is computationally tractable, it comes at the cost of significantly increased CPU times. The latter is primarily due to the explosion of the time needed to solve the pricing problem with their dynamic programming algorithm. However, they also showed that adopting the multigraph representation led to a cost reduction of up to 10.5% and 10% when compared with solutions obtained on the min-cost and min-time graphs, respectively.

To the best of our knowledge, the only study addressing an E-VRP on the road network is that of Florio et al. (2021). Their work incorporates time-dependent stochastic travel speed and energy consumption. Consequently, they treat energy consumption as a random variable and model EV autonomy as a chance constraint. An interesting aspect of their solution methods is their utilization of both the classical customer-based 1-graph and the actual road network. They initially "price" elementary routes on the minimum-cost 1-graph. This pricing is achieved through a labeling algorithm that does not rely on dominance rules or energy feasibility checks. Instead, label growth is exclusively controlled by completion bounds obtained through solving resource-constrained shortest-path problems. Once the labeling algorithm identifies minimum-cost routes with negative reduced costs, the energy feasibility of these routes is verified using a Monte Carlo method. In cases where these minimum-cost routes are proven infeasible, the algorithm proceeds to find routes directly on the road network graph. Resourcefully, they streamline the search by seeking routes that visit the same customer sequences as the minimum-cost routes. Note that this work does not delve into the EV scheduling but rather explores the impact of traffic congestion on energy consumption.

In our research, we aim to bring together the three streams of research discussed above. We augment on-site EV charging scheduling problems by considering, for the first time, the routing component. We also use a multigraph to better reflect on the trade-offs between energy consumption and travel distance/time on road networks. As a result, our modeling approach is flexible enough to arbitrate between cost efficiency of the routes and their energy consumption. This flexibility is especially crucial to unlocking the on-site charging scheduling component when charging infrastructure is limiting (as is common in practice).

3. Problem Statement

Let $\mathcal{C} = \{1, 2, \ldots, C\}$ be a set of C customers that need to be served. We define the mE-VRSPTW on a directed multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{0\} \cup \mathcal{C}$ is the set of vertices, vertex 0 represents the depot, and $\mathcal{A} \subseteq \cup_{(i,j)\in\mathcal{V}^2} \mathcal{A}_{(i,j)}$ is the *multiset* of arcs connecting the vertices in $\mathcal{V}$. The set $\mathcal{A}_{(i,j)} = \{(i,j)^k \mid k = 1, \ldots, K_{(i,j)}\}$ represents $K_{(i,j)}$ alternative ways to travel directly from $i \in \mathcal{V}$ to $j \in \mathcal{V}$.

A sufficiently large homogeneous fleet of EVs is available to serve the customers. Every vehicle is associated with a load capacity $Q \in \mathbb{Z}_+$ and equipped with a battery storing an amount of energy between zero (a minimum amount) and a maximum amount $E \in \mathbb{Z}_+$. All vehicles may depart from the depot as early as time $\underline{T} \in \mathbb{Z}_+$ and must return to the depot no later than time $\overline{T} \in \mathbb{Z}_+$, where $\underline{T} < \overline{T}$.

Every vertex $i \in \mathcal{V}$ is associated with a load $q_i \in \{0, 1, \ldots, Q\}$ that must be picked up, and a time window $[\underline{t}_i, \overline{t}_i]$ during which the vehicle visiting that vertex must start service, where $\underline{t}_i, \overline{t}_i \in [\underline{T}, \overline{T}]$ and $\underline{t}_i \le \overline{t}_i$. The EVs can arrive at a customer location before the time window opening but must wait to begin service. For the depot, we set $q_0 = 0$ and $[\underline{t}_0, \overline{t}_0] = [\underline{T}, \overline{T}]$.

When traveling along an arc $(i,j)^k \in \mathcal{A}$, a vehicle incurs a driving distance (or travel cost) $c_{(i,j)^k} \ge 0$, a travel time $t_{(i,j)^k} \ge 0$ (including the service time at vertex i if any), and an energy consumption $e_{(i,j)^k} \in \{0, 1, \ldots, E\}$. Considering that there are multiple ways to travel between two locations, $\underline{c}_{(i,j)}$, $\underline{t}_{(i,j)}$, and $\underline{e}_{(i,j)}$ denote the minimum distance, travel time, and energy consumption, respectively, when traveling from $i \in \mathcal{V}$ to $j \in \mathcal{V}$. We assume that these minimum values satisfy the triangle inequality.

At the depot, there are $B \in \mathbb{Z}_+$ homogeneous chargers available. Thus, at most B vehicles can charge simultaneously. The vehicles may start charging at time 1 (that is, $\underline{T} - 1$ units of time prior to the depot's opening) and must stop charging before the starting time of their assigned route. This means that a vehicle may delay its departure from the depot to have more time available to recharge. Accordingly, we define the set of periods at which the vehicles may charge as $\mathcal{T} = \{1, \ldots, \underline{T}, \ldots, \tilde{T} - 1\}$, where a period $t \in \mathcal{T}$ is seen as a period spanning the semiopen interval $[t, t+1[$ and is indexed (identified) by its start time. Moreover, $\tilde{T} - 1$ is the last period in which a vehicle may charge given that $\tilde{T} = \lfloor \max_{j\in\mathcal{C}} \{\overline{t}_j - \underline{t}_{(0,j)}\} \rfloor$ is the latest time a vehicle can depart from the depot to service a customer. Depending on the context, we also refer to $\mathcal{T}$ as the set of charging period vertices representing these periods. As shown in Figure 1, the charging scheduling is a discrete-time problem where decisions are made at the start of each period. In contrast, routing is a continuous-time problem defined over the interval $[\underline{T}, \overline{T}]$.

At the start of the planning horizon (i.e., at time 1), each vehicle begins with zero energy. This assumption is not restrictive, as a zero energy level does not indicate an empty battery but rather a minimal state of charge. The battery is recharged according to a concave piecewise-linear function f with $P \in \mathbb{N}$ pieces, where $\mathcal{P} = \{1, \ldots, P\}$ is

Figure 1. Routing and Charging Scheduling Time Horizons

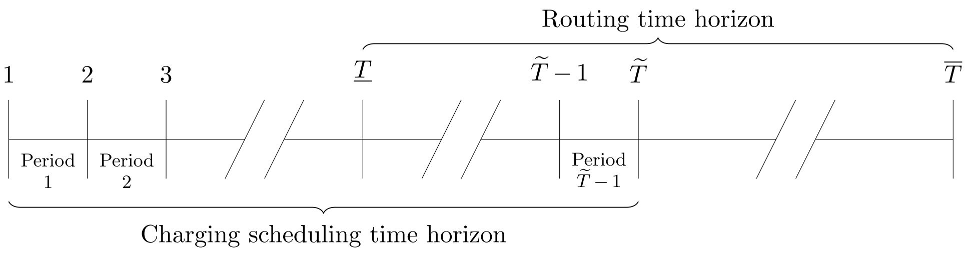
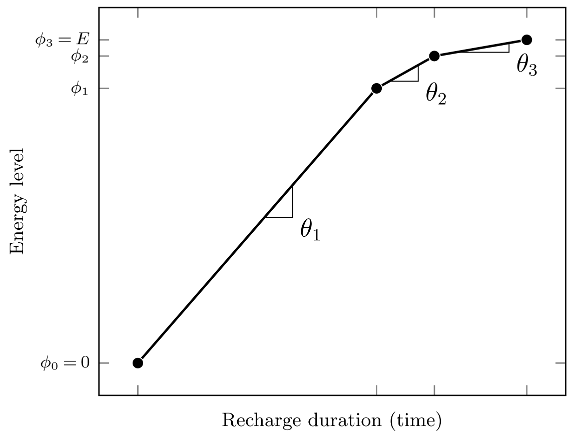


Figure 2. Amount of Energy Charged as a Function of the Duration Spent Recharging, According to Montoya et al. (2017)



the ordered set of pieces. Each piece $p \in \mathcal{P}$ begins at an energy level ϕ_{p-1} and ends at an energy level ϕ_p, satisfying $0 = \phi_0 < \phi_1 < \cdots < \phi_P = E$. Associated with each piece $p \in \mathcal{P}$, there is a recharging rate $\theta_p > 0$ (in energy per period) such that $\theta_1 > \theta_2 > \cdots > \theta_P > 0$. Figure 2 presents a three-piece piecewise-linear function. Given an energy level $e \in \{0, 1 \ldots, E\}$, the inverse recharging function $f^{-1}(e)$ maps to the recharge duration required to reach this level, assuming that the EV initially has zero energy. This function can be computed as

$$f^{-1}(e) = \sum_{p=1}^{p'} \frac{1}{\theta_p}(\min\{e, \phi_p\} - \phi_{p-1}), \tag{1}$$

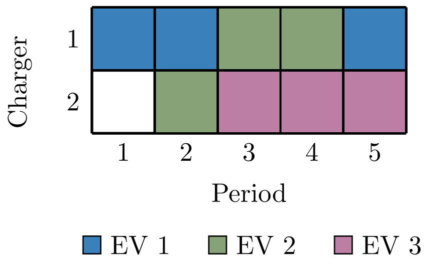
where $p' \in \mathcal{P}$ is the piece encompassing energy level e, that is, $\phi_{p'-1} \le e \le \phi_{p'}$. We can precompute Function (1) given that energy capacity and energy consumption are assumed to be integer. This assumption is not restrictive as any energy quantity can be approximated using scaled integer values. Furthermore, an EV can charge its battery only once and at a single charger. Figure 3 shows a charging schedule with two chargers, three EVs, and five periods that violates these two conditions. In this schedule, EV 1 (blue) interrupts its recharge to share a charger, while EV 2 (green) relocates to a different charger.

The mE-VRSPTW consists in routing the fleet of EVs and scheduling their preceding overnight charging such that (1) each customer is visited exactly once by a single vehicle and within its time window, (2) each route starts and ends at the depot, (3) each route is load- and energy-feasible, (4) the number of EVs simultaneously charging at the depot is at most the number of available chargers, and (5) a vehicle receives no more than one continuous recharge (preemption is prohibited) and in a single charger. Condition (5) helps preserve the health of the batteries, because frequent interruptions during the charging process accelerate their degradation (Gan et al. 2012, Sun et al. 2018). The objective of the mE-VRSPTW is to minimize the total distance traveled by the EVs. We summarize the notation for the problem statement in Table A.1 in Appendix A.

3.1. Illustrative Example

Figure 4 illustrates a feasible solution for a small mE-VRSPTW instance involving $C = 4$ customers and two EVs. Each EV has a load capacity of $Q = 10$ and is equipped with a battery capable of storing a maximum amount of

Figure 3. (Color online) Infeasible Charging Schedule



energy of $E = 8$. The load associated with each customer $i \in \{1,2,3,4\}$ is $q_i = 5$. Figure 4(a) shows the time window associated with each vertex and the distance, travel time, and energy consumption associated with each used arc and some of the unused arcs. With the numerical values shown, $\tilde{T} = 19 - 2 = 17$ because of customer 3. There is $B = 1$ charger located at the depot, and the recharging process occurs according to the piecewise-linear function depicted in Figure 4(b).

Figure 4(a) presents a feasible routing solution. EV 1 visits customers 1 and 2 (blue arcs), and EV 2 visits customers 3 and 4 (green arcs). The EVs do not use the gray arcs, but these show several ways to travel between two locations. By leaving the depot at times 7 and 16, respectively, EV 1 and EV 2 service their customers within their time window. Note that EV 1 always arrives early at the customers' location and thus waits to begin service. Both EVs carry a load of 10, satisfying the load capacity. Figure 4(b) presents a feasible charging schedule. EV 1 charges during six periods, which translates to over five units of energy. Subsequently, EV 2 charges during 9 periods, resulting in 7.5 units of energy. Thus, both EVs have sufficient energy to complete their routes. Note that in the mE-VRSPTW, the chosen routes significantly impact the charging scheduling problem. For instance, replacing the arc (1, 2) with attributes (4, 4, 1) in the route for EV 1 by that with attributes (2, 2, 5) would yield a lower-cost route that is, however, energy-infeasible.

3.2. Set-Partitioning Formulation

We model the mE-VRSPTW as a route-based set-partitioning problem, where a route is defined by the customers that it visits, the arcs that it traverses, and the periods during which the associated EV charges. Let $a_i^r \in \{0,1\}$ be a parameter indicating whether route r visits customer $i \in \mathcal{C}$, $a^r_{(i,j)^k} \in \{0,1\}$ a parameter indicating whether route r traverses arc $(i,j)^k \in \mathcal{A}$, and $b_t^r \in \{0,1\}$ a parameter indicating whether route r charges during period $t \in \mathcal{T}$. The cost c_r of a route r corresponds to its total distance traveled, that is,

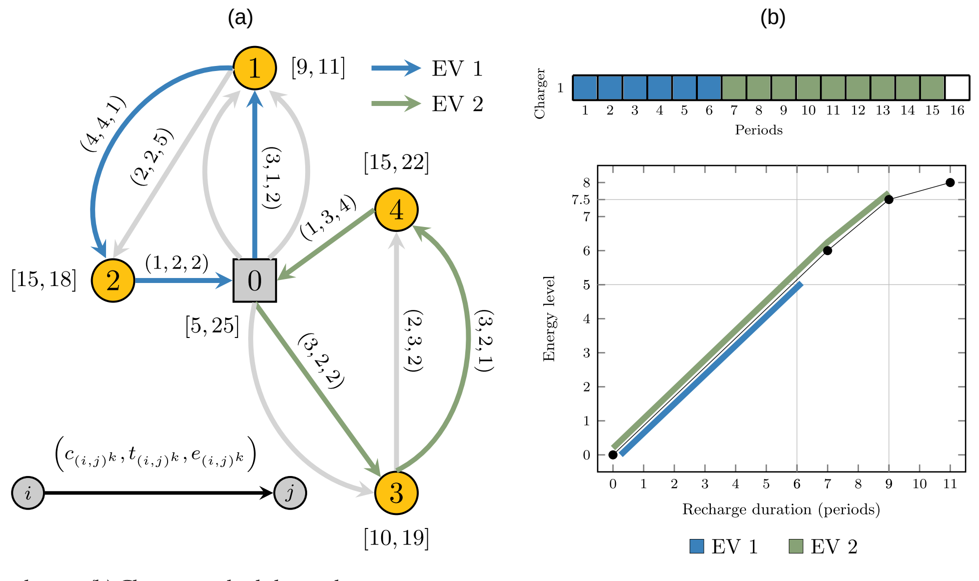
$$c_r = \sum_{(i,j)^k \in \mathcal{A}} c_{(i,j)^k} a^r_{(i,j)^k}. \tag{2}$$

For a route r to be load-feasible, it must hold that

$$\sum_{i \in \mathcal{C}} q_i a_i^r \leq Q.$$

For a route to be time-feasible, it must arrive at its visiting vertices prior to their time window closing. Finally, for a route r to be energy-feasible, its charge must be conducted nonpreemptively, and its energy level when

Figure 4. (Color online) Example of a Feasible Solution to the mE-VRSPTW



Notes. (a) Routing solution. (b) Charging scheduling solution.

departing the depot must be sufficient, namely,

$$f^{-1}\left(\sum_{(i,j)^k\in\mathcal{A}} e_{(i,j)^k} a^r_{(i,j)^k}\right) \le \sum_{t\in\mathcal{T}} b^r_t. \tag{3}$$

Let $\mathcal{R}$ be the set of all possible routes satisfying load, time window, and energy constraints and $\lambda_r \in \{0,1\}$ be a variable indicating whether route $r\in\mathcal{R}$ is selected in the solution. Our formulation of the mE-VRSPTW as a set-partitioning problem is

$$\min \sum_{r\in\mathcal{R}} c_r\lambda_r \tag{4a}$$

$$\text{s.t.} \sum_{r\in\mathcal{R}} a^r_i\lambda_r = 1 \qquad \forall i\in\mathcal{C}, \tag{4b}$$

$$\sum_{r\in\mathcal{R}} b^r_t\lambda_r \le B \qquad \forall t\in\mathcal{T}, \tag{4c}$$

$$\lambda_r \in \{0,1\} \qquad \forall r\in\mathcal{R}. \tag{4d}$$

The objective (4a) minimizes the total routing cost. Constraints (4b) guarantee that every customer is visited exactly once by a single EV. Constraints (4c) enforce the depot chargers' capacity. Finally, Constraints (4d) establish the binary domain of the variables. In this model, the routing and charging scheduling components of the mE-VRSPTW define the parameters a^r_i and b^r_t. They are individually restricted by Constraints (4b) and (4c), but also jointly by the route feasibility constraints hidden in the definition of set $\mathcal{R}$.

Because there is potentially a large number of routes in $\mathcal{R}$, Model (4) is typically too large to be solved using a MILP solver. Instead, we dynamically add variables and valid inequalities using a BPC algorithm. We refer to Model (4) as the *integer master problem*.

4. Branch-Price-and-Cut Algorithm

This section presents a BPC algorithm to solve the integer master problem (4). A BPC algorithm is a branch-and-bound (BB) algorithm in which CG computes the lower bounds, and cutting planes are incorporated to strengthen the linear relaxations in the search tree. A linear relaxation of the integer master problem is called a master problem (MP). To solve it, the CG algorithm iterates between an MP restricted to a subset of route variables, the so-called restricted master problem (RMP; see Section 4.1), and a pricing problem (PP; see Section 4.2). Given a subset of routes, the RMP determines the proportion of each route in a candidate solution. The role of the PP is to find promising and feasible routes (including their charging schedules) to add to the RMP or prove that none exists. For this reason, the PP handles the intraroute constraints—namely, the load, time window, and energy requirements—and is solved using a labeling algorithm (see Section 4.3). Once the PP cannot find an improving route, the current RMP solution yields an optimal solution to the MP. The cost of this (possibly fractional) solution provides a lower bound on the optimal value. When the solution found is fractional, valid inequalities can be incorporated to strengthen the linear relaxation and obtain better lower bounds (see Section 4.4), or integrality can be enforced by branching (see Section 4.5). Otherwise, the routes (and charging schedules) selected by the MP form a feasible solution to Model (4). We present the outline and setup of the BPC algorithm in Section 4.6 and summarize its notation in Table B.1 in Appendix B.

4.1. The Restricted Master Problem

The MP at the root node of the BB search tree is the linear relaxation of Model (4). Together, Constraints (4b) and $\lambda_r \ge 0$ ensure that $0 \le \lambda_r \le 1$ for all $r\in\mathcal{R}$. During the BB search, valid inequalities and branching decisions are incorporated into the linear program.

In a CG iteration, the RMP is obtained by replacing the set $\mathcal{R}$ in the MP with a relatively small subset $\mathcal{R}'\subseteq\mathcal{R}$. Solving the RMP yields a primal solution and a complementary dual solution $(\alpha_i)_{i\in\mathcal{C}}, (\beta_t)_{t\in\mathcal{T}}$, where $\alpha_i\in\mathbb{R}$ is the dual variable associated with Constraint (4b) indexed by $i\in\mathcal{C}$ and $\beta_t \le 0$ is the dual variable associated with Constraint (4c) indexed by $t\in\mathcal{T}$. Extending this primal solution to the MP by setting $\lambda_r = 0$ for all $r\in\mathcal{R}\setminus\mathcal{R}'$ leads to an optimal MP solution if the reduced cost of every route $r\in\mathcal{R}\setminus\mathcal{R}'$ is nonnegative. Therefore, using the dual solution, the PP tries to find routes with a negative reduced cost. When such routes are found, they are added to the subset $\mathcal{R}'$ in the RMP, and the CG algorithm iterates. Otherwise, the CG algorithm stops.

4.2. The Pricing Problem

The PP is responsible for identifying, or *pricing*, routes to be incorporated into the RMP. To achieve this, it considers the routing and charging schedules for an individual vehicle while managing intraroute constraints. Given a dual solution to the RMP, the PP consists in identifying routes with negative reduced cost, that is, routes $r \in \mathcal{R}$ such that

$$c_r - \sum_{i \in \mathcal{C}} \alpha_i a_i^r - \sum_{t \in \mathcal{T}} \beta_t b_t^r < 0. \tag{5}$$

It is defined on a directed multigraph $\mathcal{G}_P = (\mathcal{V}_P, \mathcal{A}_P)$ (see Figure 5). The set of vertices is $\mathcal{V}_P = \{s\} \cup \mathcal{T} \cup \{\underline{0}\} \cup \mathcal{C} \cup \{\overline{0}\}$, where s is a dummy source vertex, $\underline{0}$ and $\overline{0}$ are the source and sink depot vertices, respectively, and there is a vertex representing each charging period $t \in \mathcal{T}$ and each customer $i \in \mathcal{C}$. The set of arcs is $\mathcal{A}_P = \mathcal{A} \cup \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$, where

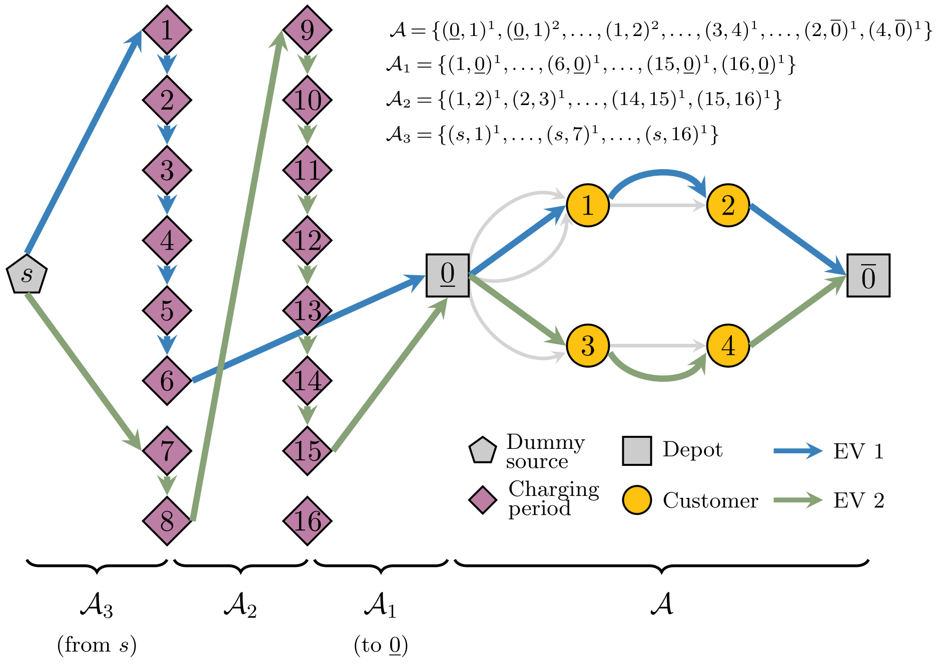
- $\mathcal{A} = \{(\underline{0}, j)^k | j \in \mathcal{C}, k = 1, \ldots, K_{(\underline{0},j)}\} \cup \{(i, \overline{0})^k | i \in \mathcal{C}, k = 1, \ldots, K_{(i,\overline{0})}\} \cup \{(i,j)^k \in \mathcal{C} \times \mathcal{C} | i \neq j,\ q_i + q_j \leq Q, \underline{t}_i + t_{(i,j)^k} \leq \overline{t}_j, \underline{e}_{(\underline{0},i)} + e_{(i,j)^k} + \underline{e}_{(j,\overline{0})} \leq E, k = 1, \ldots, K_{(i,j)}\}$ is the multiset of arcs connecting the vertices of $\mathcal{V} = \{\underline{0}\} \cup \mathcal{C} \cup \{\overline{0}\}$, where condition $\underline{e}_{(\underline{0},i)} + e_{(i,j)^k} + \underline{e}_{(j,\overline{0})} \leq E$ ensures that the arc $(i,j)^k$ can be traversed in the most optimistic energy-consumption case,
- $\mathcal{A}_1 = \{(t, \underline{0})^k | t \in \mathcal{T}, k = 1\}$ is the set of arcs connecting charging period vertices with the source depot vertex,
- $\mathcal{A}_2 = \{(t, t+1)^k | t \in \mathcal{T}, t < \tilde{T} - 1, k = 1\}$ is the set of arcs connecting successive charging period vertices, and
- $\mathcal{A}_3 = \{(s, t)^k | t \in \mathcal{T}, k = 1\}$ is the set of arcs connecting the dummy source with charging period vertices.

For $i \in \mathcal{V}_P \setminus \mathcal{V}$, we set $q_i = 0$, $\underline{t}_i = \underline{T}$, and $\overline{t}_i = \overline{T}$. For $(i,j)^1 \in \mathcal{A}_P \setminus \mathcal{A}$, we set $c_{(i,j)^1} = t_{(i,j)^1} = e_{(i,j)^1} = 0$. With these definitions, we assign a modified cost $r_{(i,j)^k}$ to each arc $(i,j)^k \in \mathcal{A}_P$:

$$r_{(i,j)^k} = \begin{cases} c_{(i,j)^k} - \alpha_i, & (i,j)^k \in \mathcal{A}, \\ -\beta_i, & (i,j)^k \in \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3, \end{cases} \tag{6}$$

where $\alpha_{\underline{0}} = 0$ and $\beta_s = 0$ for notation conciseness. Figure 5 shows the routes in the solution depicted in Figure 4

Figure 5. (Color online) Illustrative Pricing Problem Multigraph


$\mathcal{A} = \{(\underline{0},1)^1, (\underline{0},1)^2, \ldots, (1,2)^2, \ldots, (3,4)^1, \ldots, (2,\overline{0})^1, (4,\overline{0})^1\}$
$\mathcal{A}_1 = \{(1,\underline{0})^1, \ldots, (6,\underline{0})^1, \ldots, (15,\underline{0})^1, (16,\underline{0})^1\}$
$\mathcal{A}_2 = \{(1,2)^1, (2,3)^1, \ldots, (14,15)^1, (15,16)^1\}$
$\mathcal{A}_3 = \{(s,1)^1, \ldots, (s,7)^1, \ldots, (s,16)^1\}$
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
s
$\underline{0}$
$\overline{0}$
1
2
3
4
Dummy source
Depot
EV 1
Charging period
Customer
EV 2
$\mathcal{A}_3$ (from s)
$\mathcal{A}_2$
$\mathcal{A}_1$ (to $\underline{0}$)
$\mathcal{A}$

mapped to the corresponding multigraph $\mathcal{G}_{\text{P}}$ of the PP. For the sake of clarity, some of the arcs in $\mathcal{A}_{\text{P}}$ are not depicted, but the arc subsets are partially written out. Note that a path from s to $\overline{0}$ passing through $\underline{0}$ defines a route (and its charging schedule).

The PP is an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) from the dummy source s to the depot sink $\overline{0}$, where the cost of an arc $(i,j)^k \in \mathcal{A}_{\text{P}}$ is given by (6). Because solving the ESPPRC is NP-hard in a strong sense (Dror 1994), researchers have explored various relaxations of the elementarity requirements when addressing the PP (see Costa et al. 2019). Elementarity can be relaxed in the PP because of Constraints (4b), which prevent the presence of nonelementary routes in integer solutions.

Among the possible PP relaxations, we have opted for the ng-Shortest Path Problem with Resource Constraints (ng-SPPRC (Baldacci et al. 2011)) that strikes a good balance between computational effort and lower bound quality. In this relaxation, each customer $i \in \mathcal{C}$ has a neighborhood $\mathcal{N}_i \subseteq \mathcal{C}$ containing the $\Delta \in \mathbb{Z}_+$ closest customers to i (in terms of distance, time, or energy) and customer i itself. An ng-path allows a cycle starting and ending at vertex $j \in \mathcal{C}$ if and only if there exists a vertex $i \in \mathcal{C}$ in the cycle for which $j \notin \mathcal{N}_i$. Therefore, such a cycle is prohibited if and only if $j \in \mathcal{N}_i$ for every vertex $i \in \mathcal{C}$ it contains. Because nonelementary routes can be found when solving this relaxation, rather than binary, we define $a_i^r \in \mathbb{Z}_+$ and $a_{(i,j)^k}^r \in \mathbb{Z}_+$ as integers. Naturally, these cycling restrictions only apply to customer vertices.

Along the lines of Contardo et al. (2015), we initialize the sets $\mathcal{N}_i, i \in \mathcal{C}$ with a relatively small number Δ of closest customers. Then, if the PP only finds nonelementary routes, the neighborhoods of the customers in every single cycle are enlarged by adding the vertex where the cycle starts. We repeat this process until an elementary route with a negative reduced cost is found or a maximum neighborhood size $\overline{\Delta}$ is reached.

4.3. Labeling Algorithm

To solve the ng-SPPRC, we employ a backward labeling algorithm, as described in Irnich and Desaulniers (2005). This algorithm constructs feasible paths in $\mathcal{G}_{\text{P}}$ starting from the sink vertex $\overline{0}$ and moving toward the source vertex s. It begins with a subpath containing only the depot sink vertex $\overline{0}$ at time $\overline{T}$. During each iteration, the method extends an existing subpath by considering all incoming arcs at its current vertex, thereby creating additional subpaths. The algorithm evaluates these subpaths until it identifies at least one complete path (i.e., a path running from the dummy source s to the depot sink vertex $\overline{0}$) with a negative reduced cost. Such a path represents a feasible route (along with its associated charging schedule) that can be included in the set $\mathcal{R}'$ in the RMP.

Every subpath between an intermediate vertex $i \in \mathcal{V}_{\text{P}}$ and the depot sink vertex $\overline{0}$ is encoded by a *label* ℓ_i. This label stores the cumulative reduced cost $r(\ell_i)$, the remaining load capacity $q(\ell_i)$, the latest service start time $t(\ell_i) \in [\underline{T}, \overline{T}]$ at vertex i that ensures time windows' feasibility, the remaining energy capacity $e(\ell_i)$, and the minimum number of charging periods $b(\ell_i)$ required to traverse the subpath. For each customer $c \in \mathcal{C}$, it also stores a resource $\mathbb{I}_c^u(\ell_i) \in \{0,1\}$ indicating whether customer c is unreachable in any extension of the subpath because of resource limitations and a resource $\mathbb{I}_c^{ng}(\ell_i) \in \{0,1\}$ indicating whether visiting customer c violates the ng-path cycling restrictions. When a resource consumption is infeasible or does not comply with the ng-path cycling restrictions, the corresponding label is discarded. To avoid enumerating all feasible routes, the labeling algorithm discards nonuseful subpaths verifying a dominance rule between labels.

To initialize the labeling algorithm, we create a label $\ell_{\overline{0}}$ representing a subpath at the sink depot vertex with $r(\ell_{\overline{0}}) = 0$, $q(\ell_{\overline{0}}) = Q$, $t(\ell_{\overline{0}}) = \overline{T}$, $e(\ell_{\overline{0}}) = E$, $b(\ell_{\overline{0}}) = 0$, and $\mathbb{I}_c^u(\ell_{\overline{0}}) = \mathbb{I}_c^{ng}(\ell_{\overline{0}}) = 0$ for all $c \in \mathcal{C}$. At each iteration, the algorithm selects an unprocessed label ℓ_j at a node j and extends it backwardly along every incoming arc $(i,j)^k \in \mathcal{A}_{\text{P}}$ to obtain new labels. The backward search identifies both the latest possible departure time from the depot and the charging duration necessary for completing the route before extending the labels to the charging period vertices. It also schedules the charge as closely as possible to its departure time from the depot to prevent battery degradation (i.e., calendar aging).

Algorithm 1 presents the label extension procedure. Lines 1–3 check whether the extension is feasible. When i is a customer vertex, line 1 discards the label if that customer is unreachable or visiting it violates the ng-path cycling restrictions. An arc traversed from a charging period vertex to the depot source vertex determines the charging end time. Therefore, line 1 discards the label when such time either is insufficient to charge the necessary energy or is greater than the latest time for starting the route. Lastly, line 1 verifies that once a label reaches the dummy source vertex, the battery is sufficiently charged.

Algorithm 1 (Label Extension)

```
Input: $\ell_j$, label representing a subpath starting at vertex $j$; $(i,j)^k \in \mathcal{A}_P$, arc along which the extension is performed.
Output: $\ell_i$, label representing a subpath starting at vertex $i$.
/* check whether the extension is feasible */
1  if $i \in \mathcal{C} \wedge (\mathbb{I}^u_i(\ell_j) = 1 \vee \mathbb{I}^{ng}_i(\ell_j) = 1)$ then exit
2  if $i \in \mathcal{T} \wedge j = \underline{0} \wedge (i < b(\ell_j) \vee i \geq t(\ell_j))$ then exit
3  if $i = s \wedge b(\ell_j) > 0$ then exit
/* update the resource consumptions */
4  $r(\ell_i) \leftarrow r(\ell_j) + r_{(i,j)^k}$
5  $q(\ell_i) \leftarrow q(\ell_j) - q_i$
6  $t(\ell_i) \leftarrow \min\{t(\ell_j) - t_{(i,j)^k}, \bar{t}_i\}$
7  $e(\ell_i) \leftarrow e(\ell_j) - e_{(i,j)^k}$
8  if $i \in \mathcal{T}$ then $b(\ell_i) \leftarrow b(\ell_j) - 1$
9  else $b(\ell_i) \leftarrow \lceil f^{-1}(E - e(\ell_i)) \rceil$
/* check whether the extension is actually feasible */
10 if $t(\ell_i) < \underline{t}_i \vee e(\ell_i) < 0 \vee b(\ell_i) \geq t(\ell_i) \vee b(\ell_i) < 0$ then exit
/* mark customers that are unreachable by resource or cycling constraints */
11 if $i \in \mathcal{C}$ then
12     for $c \in \mathcal{C}$ do
           /* unreachable customers */
13         $\mathbb{I}^u_c(\ell_i) \leftarrow \mathbb{I}^u_c(\ell_j)$
14         if $c \neq i \wedge \mathbb{I}^u_c(\ell_i) = 0 \wedge (q(\ell_i) - q_c < 0 \vee t(\ell_i) - \underline{t}_{(c,i)} < \underline{t}_c \vee e(\ell_i) - \underline{e}_{(c,i)} < 0)$ then
15             $\mathbb{I}^u_i(\ell_i) \leftarrow 1$
           /* ng-path cycling restrictions */
16         if $c = i \vee (\mathbb{I}^{ng}_c(\ell_j) = 1 \wedge c \in \mathcal{N}_i)$ then $\mathbb{I}^{ng}_c(\ell_i) \leftarrow 1$
17         else $\mathbb{I}^{ng}_c(\ell_i) \leftarrow 0$
18 return $\ell_i$
```

Lines 4–9 compute the resource consumptions. Lines 4–7 update, respectively, the subpath's reduced cost, remaining load capacity, latest time for starting service, and remaining energy capacity. A unit time charge occurs when the extension is to a charging period vertex (line 8). Otherwise, line 9 computes the number of periods required to charge according to the consumed energy and the piecewise-linear recharging function. We round up the required charging period because we have a discrete-time scheduling problem whereas the inverse recharging function could map to a noninteger value. This rounding up ensures that the route is still energy-feasible and leads to a more efficient dominance rule, as shown later.

Once the resource consumptions have been computed, line 10 checks whether the label extension is actually feasible. Specifically, when the latest time for starting service is earlier than the time window opening, the energy capacity has been exceeded, the necessary charge duration is greater than the remaining time available, or the time required to charge has already been completed, the label is discarded.

When the extension is to a customer vertex, lines 11–17 mark customers that are unreachable in any extension of the subpath because of resource limitations or cycling restrictions. Lines 13 and 15 update the unreachable customers by using lower bounds on the travel time and energy consumption (their minimum values). However, a customer not marked as unreachable might be unreachable when traveling along a specific arc. Thus, once the label is extended, we need to check if the extension is actually feasible (as in line 10). Afterward, lines 16 and 17 identify the customers that cannot be visited because of the *ng*-path restrictions. A customer can be visited only if it has not been visited before or does not belong to the neighborhood of a vertex visited after its last visit. Finally, line 1 returns a new, extended label associated with node *i*.

When a feasible label extension is found by Algorithm 1, the labeling method verifies dominance between labels associated with the same vertex to discard nonuseful subpaths. On the one hand, if an existing label dominates the new label, the new label is discarded. On the other hand, if the new label dominates an existing label, the existing label is discarded. In the case that ℓ_i and ℓ'_i dominate each other, we keep one of them to guarantee the exactness of the method. In our labeling algorithm, the dominance rule depends on whether the label's vertex is a customer (Definition 1) or a charging period (Definition 2).

Definition 1. At a customer vertex $i \in \mathcal{C}$, a label ℓ_i dominates another label ℓ_i' if the following conditions hold:

$$r(\ell_i) \le r(\ell_i'), \tag{7a}$$

$$q(\ell_i) \ge q(\ell_i'), \tag{7b}$$

$$t(\ell_i) \ge t(\ell_i'), \tag{7c}$$

$$e(\ell_i) \ge e(\ell_i'), \tag{7d}$$

$$\max\{\mathbb{I}_c^u(\ell_i), \mathbb{I}_c^{ng}(\ell_i)\} \le \max\{\mathbb{I}_c^u(\ell_i'), \mathbb{I}_c^{ng}(\ell_i')\} \quad \forall c \in \mathcal{C}. \tag{7e}$$

Conditions (7) imply that any feasible extension of the dominated label ℓ_i' is also a feasible extension of the dominant label ℓ_i with an equal or better reduced cost.

Definition 2. At a charging period vertex $i \in \mathcal{T}$, a label ℓ_i dominates another label ℓ_i' if the following conditions hold:

$$r(\ell_i) \le r(\ell_i'), \tag{8a}$$

$$b(\ell_i) \le b(\ell_i'). \tag{8b}$$

At a charging period vertex, dominance is verified only by the reduced cost and periods required to charge because the load, time, energy, unreachable customers, and cycling restrictions become irrelevant. Finally, Condition (8b) is much more likely to be satisfied when the charging period is defined in terms of the number of periods required to charge (an integer value) rather than the charge duration as defined by the inverse recharging function (1) (a nonnegative value). In any case, as we have a discrete-time scheduling problem, EVs can only charge during the entirety of a period.

4.3.1. Heuristic Labeling.

In the CG algorithm, solving exactly the PP is only necessary to prove the optimality of the current RMP solution for the MP. Consequently, one or several heuristic algorithms can be invoked first, hoping to rapidly find negative reduced cost routes (Desaulniers et al. 2008). The exact labeling algorithm is invoked only when the heuristic algorithms fail to find a negative reduced cost route.

From the (exact) labeling algorithm explained above, we derive three heuristic labeling algorithms by implementing one or both of the following two modifications. First, to alleviate the computational demand of handling a multigraph, for every pair of vertices $i, j \in \mathcal{V}$ such that $K_{(i,j)} \ge 2$, we only keep the arc $(i,j)^k \in \mathcal{A}_{(i,j)}$ with the lowest cost $c_{(i,j)^k}$. Second, we apply a simpler (and heuristic) dominance rule than stated in Definition 1 which disregards Condition (7e).

4.4. Valid Inequalities

Reinforcing the MP with valid inequalities significantly enhances the performance of branch-and-price algorithms (Costa et al. 2019). The idea is to incorporate cutting planes into the linear relaxations encountered in the search tree to obtain tighter lower bounds and reduce the size of the search tree.

First, we strengthen the MP at the root node with the following well-known rounded capacity inequality that involves all customers in $\mathcal{C}$:

$$\sum_{r \in \mathcal{R}} \lambda_r \ge \left\lceil \sum_{i \in \mathcal{C}} q_i / Q \right\rceil, \tag{9}$$

and we denote by $\pi \ge 0$ the associated dual variable. We further (dynamically) strengthen the MP with subset-row cuts (SRCs) defined over customer sets of size three (Jepsen et al. 2008). These constraints are rank-1 Chvátal-Gomory cuts derived from a subset of Constraints (4b). Given a customer triplet $S \subseteq \mathcal{C}$, $|S| = 3$, the corresponding 3-SRC is defined as

$$\sum_{r \in \mathcal{R}} \left\lfloor \frac{1}{2} \sum_{i \in S} a_i^r \right\rfloor \lambda_r \le 1, \tag{10}$$

and we denote its dual variable by $\sigma_S \le 0$. Constraint (10) specifies that, in a feasible integer solution, at most one route may serve two or more customers in S.

When CG terminates, SRCs are separated by enumerating all customer triplets and checking if the current solution violates the corresponding inequalities. Because incorporating many SRCs into the MP results in a more difficult PP, we use three rules proposed by Desaulniers et al. (2008) to select the cuts to add. First, at most $\overline{S}$ cuts can be added simultaneously, prioritizing the ones with greater violations. Second, a customer may appear in at most $\overline{C}$ of the customer triplets defining these cuts. Third, cuts are only added if the most violated one has a violation greater than or equal to a threshold $\varepsilon > 0$. We denote by $\mathcal{S}$ the subset of customer triplets for which SRCs have been added to the MP.

After incorporating into the MP the rounded capacity inequality (9) and SRCs (10) for the customer triplets in $\mathcal{S}$, the PP finds routes $r \in \mathcal{R}$ such that

$$\sum_{(i,j)^k \in \mathcal{A}_\mathrm{P}} r_{(i,j)^k} a^r_{(i,j)^k} - \sum_{S \in \mathcal{S}} \sigma_S \left\lfloor \frac{1}{2} \sum_{i \in S} a^r_i \right\rfloor - \pi < 0, \tag{11}$$

where the second term of this reduced cost (left-hand side) can be interpreted as a *penalty* σ_S that must be paid every second visit to customers in $S \in \mathcal{S}$. This penalty reflects that, in the current form, Cut (10) is nonrobust, meaning that the corresponding dual variable may not be directly incorporated into the modified arc cost. This increases the complexity of the PP and impacts the labeling algorithm. As detailed in Jepsen et al. (2008) and Desaulniers et al. (2008), for every separated SRC, a label must store a new (unrestricted) resource with the number of times (modulo 2) the corresponding subpath has visited customers in S. Then, dominance can only occur if the reduced cost of the dominant label is sufficiently less than that of the dominated label to compensate for potential extra penalizations.

4.5. Branching Rules

Because the MP is a linear program, we use branching rules to enforce integrality whenever necessary. Our BPC algorithm branches hierarchically on the number of vehicles used, the arc-flow variables, and the consecutive flow through the depot source vertex.

Branching on the number of vehicles is performed by modifying the lower bound in (9) or imposing an upper bound on its left-hand side. When an integer number of vehicles is used, we try to branch on an arc-flow variable $x_{(i,j)^k}, (i,j)^k \in \mathcal{A}_\mathrm{P}$, which has an equivalent representation in terms of the route variables $\lambda_r, r \in \mathcal{R}$. In this branching strategy, we first branch on the binary arc-flow variables $x_{(i,j)^k}, (i,j)^k \in \mathcal{A}$ (arcs between customers or a customer and the depot) by choosing one with the closest flow to 0.5. Both branching decisions, one for each child node, are imposed in the PP. To implement $x_{(i,j)^k} = 0$ (i.e., the arc must not be used in the solution), we remove from $\mathcal{R}'$ all routes using arc $(i,j)^k$ and the arc $(i,j)^k$ itself from $\mathcal{A}_\mathrm{P}$. To implement $x_{(i,j)^k} = 1$ (i.e., the arc must be used in the solution), we remove from $\mathcal{R}'$ all routes using arcs $(i,j)^l, l \neq k$—those parallel to $(i,j)^k$, arcs $(i,u)^l, u \neq j, l = 1,\dots,K_{(i,u)}$ if $i \in \mathcal{C}$, or arcs $(u,j)^l, u \neq i, l = 1,\dots,K_{(u,j)}$ if $j \in \mathcal{C}$. We also remove these arcs from $\mathcal{A}_\mathrm{P}$, forcing that, in the PP, vertex i is always visited immediately after vertex j using arc $(i,j)^k$.

When the arc-flow variables $x_{(i,j)^k}, (i,j)^k \in \mathcal{A}$, have all integer values, we branch on the arc-flow variable $x_{(i,j)^1}, (i,j)^1 \in \mathcal{A}_1 \cup \mathcal{A}_3$ for which the fractional part of its flow value $\tilde{B} \notin \mathbb{Z}_+$ is closest to 0.5. The branching is performed by imposing the following constraints in the MP: $x_{(i,j)^1} \leq \lfloor \tilde{B} \rfloor$ and $x_{(i,j)^1} \geq \lceil \tilde{B} \rceil$ but expressed in terms of the λ_r-variables. Then, the corresponding dual variable is subtracted in the modified cost of the arc $(i,j)^1 \in \mathcal{A}_1 \cup \mathcal{A}_3$.

We remark on two technical details. First, by flow conservation, when the flow through arcs in $\mathcal{A}, \mathcal{A}_1$, and $\mathcal{A}_3$ is integer, the flow throughout the entire network is also integer. For this reason, branching on arcs in $\mathcal{A}_2$ is unnecessary. Second, when branching on arcs in $\mathcal{A}_3$, the dual variable may penalize or reward the labels' reduced cost, depending on their charging start time. Thus, for Definition 2 to remain valid when such branching has been imposed, labels may be extended even after completing the charge.

Despite guaranteeing integer arc flows, the above branching decisions do not necessarily guarantee integer path flows as the flow through a charging period vertex can be larger than one. When none of these decisions can be imposed and the solution is not integer, then there exist two arcs, $(\underline{0},j)^k \in \mathcal{A}$ and $(t,\underline{0})^1 \in \mathcal{A}_1$, such that their consecutive flow

$$m_{(\underline{0},j)^k,(t,\underline{0})^1} = \sum_{r \in \mathcal{R}} a^r_{(\underline{0},j)^k} \cdot b^r_t \cdot (1 - b^r_{t+1}) \cdot \lambda_r \tag{12}$$

is fractional. We branch on these arcs as follows. On the first branch, we forbid using arc $(t,\underline{0})^1$ immediately after arc $(\underline{0},j)^k$. On the second branch, if arc $(\underline{0},j)^k$ is used, it must be immediately followed by arc $(t,\underline{0})^1$ (i.e., we forbid the use of all arcs $(t',\underline{0})^1 \in \mathcal{A}_1 \setminus \{(t,\underline{0})^1\}$). These decisions are imposed in the PP and require modifying the labeling algorithm. For the first decision, a label $\ell_{\underline{0}}$ that used arc $(\underline{0},j)^k$ cannot be extended along arc $(t,\underline{0})^1$ and, for the second decision, such a label can only be extended along this arc. For both decisions, $\ell_{\underline{0}}$ can only dominate labels that have also reached vertex $\underline{0}$ through arc $(\underline{0},j)^k$.

The branching on consecutive flow has been used for other VRPs; see, for example, Desaulniers (2010). It ensures that our algorithm is exact. However, it was unnecessary for our computational experiments because branching on the number of vehicles used and arc-flow variables was sufficient to obtain integer path flows.

4.6. Outline and Setup

Concisely, the outline of the BPC algorithm is the following:

Step 1. Choose an unprocessed node in the BB tree. If the lower bound is greater than or equal to the (global) upper bound, prune the node and repeat step 1.

Step 2. Solve the RMP.

Step 3. Solve the PP using the labeling algorithm. If routes with negative reduced cost are found, add them to the RMP and go back to step 2.

Step 4. Separate SRCs and add them to the MP. If at least one cut is separated, go back to step 2.

Step 5. If the MP solution is fractional, mark the node as processed, branch, add the two child nodes to the set of unprocessed nodes in the BB tree, and go back to step 1. Otherwise, update the upper bound (if possible) and prune the node.

In step 1, we choose the node with the lowest lower bound, that is, we use a best-first search strategy. In step 2, we use a commercial linear programming solver and retrieve the dual solution. In step 3, we allow a maximum of 400 routes to be generated. Additionally, we first solve the PP using the heuristic labeling algorithm and we only invoke the exact algorithm if the heuristic fails to find a negative reduced cost route. For the *ng*-path relaxation, we consider an initial neighborhood size of $\Delta = 7$ and a maximum neighborhood size of $\overline{\Delta} = 12$ for instances with tighter time windows (type 1 instances in Section 5.1). For instances with looser time windows (type 2 instances in Section 5.1), we set these parameters to $\Delta = 12$ and $\overline{\Delta} = 17$. To separate the SRCs in step 4, we set $\overline{S} = 30$, $\overline{C} = 5$, and $\varepsilon = 0.1$. Finally, in step 5, we use hierarchical branching rules as explained in Section 4.5. We selected the parameter values based on preliminary tests and previous research (Jepsen et al. 2008, Contardo et al. 2015, Desaulniers et al. 2019).

5. Computational Results

This section reports the computational results of our BPC algorithm for solving the mE-VRSPTW. This algorithm was implemented in Java using the Branch-and-Price framework of the Java OR library (`jORLib`) and the Java graph theory library (`JGraphT`) by Michail et al. (2020). We relied on the IBM CPLEX solver, version 22.1, for solving the RMPs. All experiments were run on a PC with an Intel Core i7-8665U CPU @ 1.90 GHz with 16 GB of RAM and subject to a two-hour computational time limit per instance. A snapshot of our code and data is available in Yamín et al. (2024).

5.1. Test Instances

To perform our experiments, we construct a set of instances from the well-known VRPTW benchmark of Solomon (1987). These instances have a 4-parameter naming convention $\mathtt{DTm\text{-}n}$. Parameter $\mathtt{D}$ is the customer geographical distribution: *R* for random, *C* for clustered, or *RC* for a mix of random and clustered. Parameter $\mathtt{T}$ is the tightness of the time windows: instances of type 1 have tighter time windows than instances of type 2. Parameter $\mathtt{m}$ is the two-digit instance number, and parameter $\mathtt{n}$ is the number of customers. We consider $\mathtt{n} = 25, 50$. These instances are derived from the 100-customer instances by considering only the first $\mathtt{n}$ customers. For each instance, the vehicles' load capacity Q, and the load q_i, time window $[\underline{t}_i, \overline{t}_i]$, and service time associated with each vertex $i \in \mathcal{C}$ are given. These instances are well suited for city logistics problems as customers are distributed over a 100- by 100-km^2 area, representing a semiurban operation. In total, our testbed is made up of 112 instances, unevenly distributed into 12 groups.

We use minutes as the time unit of discretization and kilometers as the distance unit. Furthermore, we assume that the EVs travel at a fixed speed of 60 km/h. Despite the traveling speed being slightly higher than most cities' average speed, this assumption guarantees that the time windows of Solomon's instances remain valid.

Similar to Montoya et al. (2017), the EVs in our instances are `Peugeot iOns`. This EV has an energy capacity E of 16 kWh. To capture the trade-off between conflicting resources, we consider two alternative ways to travel between each pair of customers or a customer and the depot: (1) the minimum driving distance (and travel time) alternative $(i,j)^1 \in \mathcal{A}$ and (2) the minimum energy consumption alternative $(i,j)^2 \in \mathcal{A}$. In the former, the driving distance $c_{(i,j)^1}$ is the *Euclidean distance* and the energy consumption $e_{(i,j)^1} = c_{(i,j)^1} \cdot 0.175$ kWh/km given the iOn's average consumption rate (Electric Vehicle Database 2022). In the latter, the driving distance $c_{(i,j)^2}$ is equal to the *Manhattan distance* and the energy consumption $e_{(i,j)^2} = c_{(i,j)^2} \cdot 0.125$ kWh/km. The latter is the iOn's consumption rate used by Montoya et al. (2017). With these settings, one alternative may dominate the other; in this case, we only consider the dominant alternative. Table 1 (column 3) presents the average number of alternative ways to travel between two locations for each group of instances.

The EVs are charged with a "slow" 11-kW charger, one of the most widely used in practice. Consequently, the three-piece piecewise-linear recharging function is defined by $\theta_1 = 10.79$, $\theta_2 = 5.71$, and $\theta_3 = 1.60$ kWh/h and $\phi_1 = 13.6$, $\phi_2 = 15.2$, and $\phi_3 = 16.0$ kWh (Montoya et al. 2017). Recall that EVs may start charging $\underline{T} - 1$ units of time before the depot's opening. We choose the value of $\underline{T} = 124$ so that a full charge can be conducted before the depot's opening time. In addition, Table 1 (column 4) shows the number of chargers, B, for each instance. We choose B as the minimum number of chargers such that all the instances in the same group remain feasible and solvable within the time limit. As we show later, these values yield difficult charging scheduling problems.

Our solution method assumes that the energy capacity and consumption are integer values. This assumption is needed to precompute the inverse of the recharging function (1) for each energy level $e \in \{0, 1, \dots, E\}$. Thus, we use decawatt-hours (dWh) as a unit of energy, which is equivalent to working with kWh and considering two decimal points. Also, we compute the driving distance $c_{(i,j)^k}, (i,j)^k \in \mathcal{A}$ as in Kohl et al. (1999); we obtain integer values by truncating the nonnegative distances with one decimal point and multiplying the result by 10. Although our BPC algorithm does not require this truncation, it helps with the results' reproducibility. We compute the travel time $t_{(i,j)^k}, (i,j)^k \in \mathcal{A}$ as the sum of the corresponding driving distance and the service time at vertex i.

As explained in Appendix C, we set certain energy-related parameters slightly differently in the RC 50-customer instances because of its distinctive customers' locations. Furthermore, we set the arcs' minimum distance, travel time, and energy consumption by running all-pairs shortest-path algorithms to ensure that the triangle inequality holds for these values. All instances are publicly available at www.vrp-rep.org, VRP-REP-ID: 2024-0001 (Mendoza et al. 2014).

5.2. Performance of the BPC Algorithm

Table 1 summarizes the BPC algorithm's computational performance by presenting average values for each instance group. Column 5 shows the integrality gap, computed as $\frac{\text{UB}-\text{LB}}{\text{UB}} \cdot 100$, where LB and UB refer, respectively, to the lower bound at the root node and the optimal integer value. Columns 6 and 7 provide the number of SRCs separated at the root node and the number of nodes in the BB tree, respectively. Column 8 shows the total computational time in seconds. Columns 9 and 10 report the proportion of time spent on the RMP and the PP, respectively. These proportions do not add to 100% because they do not include some steps in the BPC algorithm, such as node selection and branching. The electronic companion EC.1 presents the (disaggregated) results for each instance.

All 112 instances are solved to optimality within the time limit. The lower bounds at the root node are tight, with a maximum integrality gap of 1.86%. This result may plausibly be attributed to the many separated SRCs. The resulting gaps allow proving optimality by exploring relatively small BB trees. Still, enforcing integrality in the mE-VRSPTW can be harder than in other vehicle routing problems, given that an integer flow value of the arcs between customers does not guarantee an integer solution to this problem. This can be seen in some instance groups where the integrality gap is zero, but some branching is still necessary. In these instances, most branching

Table 1. Summary of the BPC Algorithm's Computational Performance

							Time		
Instance	C	Avg. # of altern.	B	Integrality gap (%)	Cuts	Nodes	Total (s)	RMP (%)	PP (%)
R1	25	1.64	3	0.18	13.3	3.2	1.7	12.2	71.9
C1		1.61	1	0.22	10.6	1.7	11.0	3.5	92.3
RC1		1.53	3	0.24	59.0	2.5	4.4	11.8	78.3
R2	25	1.66	1	0.00	10.4	6.3	2.9	8.8	63.8
C2		1.62	1	0.00	43.1	1.5	57.5	2.8	94.2
RC2		1.53	1	0.00	39.8	7.3	4.2	16.4	63.4
R1	50	1.63	6	0.53	92.6	21.0	201.3	7.4	87.7
C1		1.68	2	0.18	21.2	1.4	53.0	3.7	93.5
RC1		1.48	4	0.21	108.5	2.3	560.0	2.7	96.4
R2	50	1.65	2	0.15	78.4	15.5	129.2	14.3	76.9
C2		1.68	1	0.00	80.5	2.0	221.9	6.7	91.1
RC2		1.49	2	0.00	33.8	2.5	26.1	8.9	84.1

decisions are imposed over the arcs in $\mathcal{A}_1 \cup \mathcal{A}_3$, meaning that, given a fractional solution at the root node, the algorithm finds an equivalent integer solution (in terms of cost) through branching.

The BPC algorithm yields average CPU times of 12.2 seconds for 25-customer instances and 192.5 seconds for 50-customer instances. As expected, the CPU time increases exponentially with the number of customers. However, an analysis of these times categorized by instance type reveals a somewhat counterintuitive result. Typically, instances of type 2 (i.e., with wider time windows) are considered more challenging because of their larger search space in the PP. Surprisingly, the arithmetic mean of the CPU times for type 1 instances (129.8 s) and type 2 instances (72.8 s) suggests the opposite. Detailed results in the electronic companion EC.1 reveal that some type 1 instances are particularly challenging to solve. This is because narrow time windows can significantly impact the complexity of the charging scheduling problem. For instance, the average latest departure time from the depot ($\tilde{T}$) is 581.5 for type 1 instances and 1,736.5 for type 2 instances. In other words, on average, type 1 instances have less time available for completing charging operations, making the scheduling problem more complex. When looking at the same CPU times through the lens of the less-sensitive-to-outliers geometric mean (15.7 s for type 1 and 20.2 s for type 2), it becomes clear that, in general, type 2 instances are indeed harder to solve, as intuition dictates. Note also that vehicle autonomy restricts the maximum number of customers that can be serviced in the same route, reducing the difficulty to price the columns for the type 2 instances.

Table 2 presents a portrait of the characteristics of the solutions, reporting averages by instance group. Columns 4 and 5 show the number of EVs used in the optimal solution and its cost, respectively. Columns 6–8 provide information about the solution's charging schedule. Time in use (TIU) represents the percentage of time between the first and last periods during which at least one EV is actively charging and, consequently, at least one charger is in use. The *chargers in use* (CIU) is the proportion of chargers used at the depot in that time frame. Finally, the *time at capacity* (TAC) is the proportion of that time frame when all chargers are simultaneously used. Note that when $B = 1$, these three metrics have the same value. Appendix D illustrates the computation of these metrics for the optimal solution of instance RC104-25.

The average TIU in type 1 and type 2 instances is 94% and 69%, respectively. Whereas in type 1 instances the chargers are continuously used, type 2 instances exhibit a more spread-out charging pattern, including periods of charging inactivity. The pronounced temporal concentration in type 1 instances' charging schedules leads to a higher average CIU (74%) compared with those of type 2 instances (63%). Nevertheless, the charging infrastructure operates at full capacity for nearly 55% of the time for both instance types. This parity in full-capacity operation between these types is attributed to type 2 instances featuring fewer chargers, often only one. These results indicate that type 1 instances experience more congested charging infrastructure, partially because of the increased number of EVs deployed for visiting customers and the shorter charging horizons.

In conclusion, our BPC algorithm is capable of efficiently solving to optimality instances with up to 50 customers. These results are particularly noteworthy given that the mE-VRSPTW combines a continuous-time VRP on a multigraph with a discrete-time charging scheduling problem, both of which are already complex when solved independently.

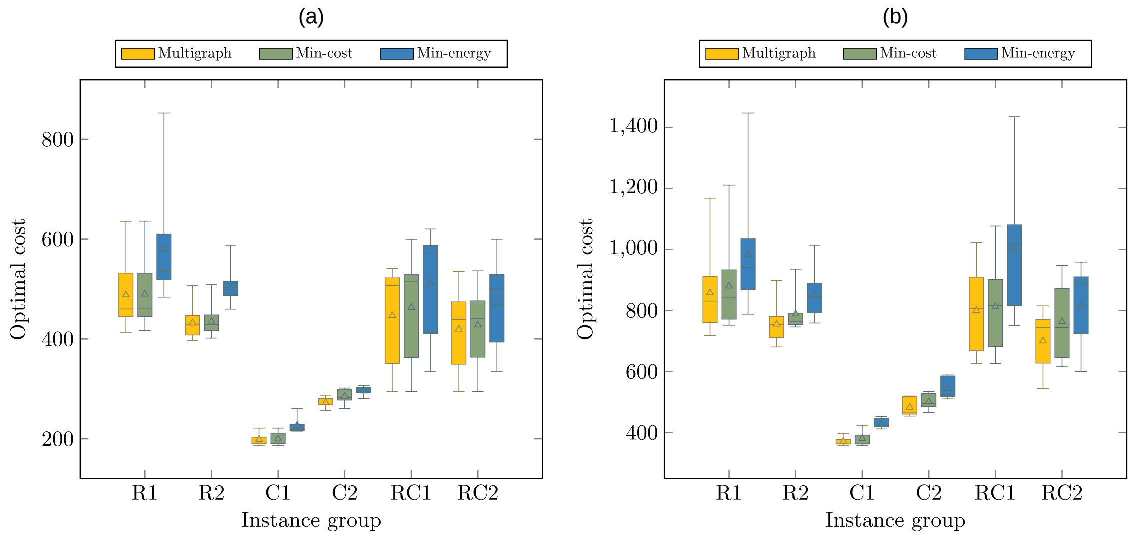
5.3. Benefits of the Multigraph Representation

One key element in the mE-VRSPTW is the possibility of exploiting a multigraph to optimize trade-offs between travel distance/time and energy consumption. In this section, we present an experiment assessing the benefits of this

Table 2. Summary of the Solution Characteristics

Instance	C	B	EVs used	Total cost	Time in use (%)	Chargers in use (%)	Time at capacity (%)
R1	25	3	6.0	488.32	94.9	73.0	49.6
C1		1	3.2	198.61	75.3	75.3	75.3
RC1		3	5.0	446.56	100.0	75.1	47.4
R2	25	1	4.8	431.57	74.3	74.3	74.3
C2		1	3.0	272.46	37.9	37.9	37.9
RC2		1	4.6	419.40	87.2	87.2	87.2
R1	50	6	10.0	857.63	100.0	72.9	40.4
C1		2	5.2	371.03	92.1	76.9	61.6
RC1		4	7.6	800.14	99.6	74.5	54.1
R2	50	2	8.1	755.05	86.8	70.8	54.7
C2		1	5.4	482.46	34.2	34.2	34.2
RC2		2	6.4	699.88	87.7	66.9	46.1

Figure 6. (Color online) Impact of Multigraph Representation on Optimal Cost

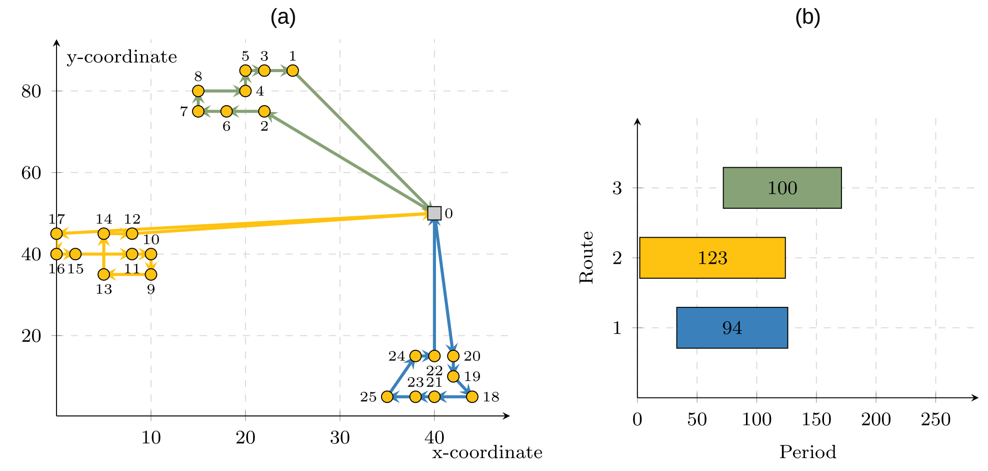


Notes. (a) Twenty-five-customer instances. (b) Fifty-customer instances.

modeling feature. We created two additional copies of the 112 instances that are defined over 1-graphs. In the first copy, we retained only the minimum driving distance arcs (min-cost graph), and in the second, we kept only the minimum energy consumption arcs (min-energy graph). The remaining data (e.g., demands, customer locations, and time windows) remain unchanged. Note that because we assume a traveling speed of 60 km/h and the travel cost is proportional to the driving distance, the min-cost and min-time graphs are equivalent. Therefore, we did not consider the latter in this experiment. We solved all 336 instances (3×112) using our BPC algorithm and found 335 optimal solutions although instance RC101-50 on the min-cost graph was deemed infeasible. Figure 6 displays box plots of the optimal costs for these three instance sets. They indicate the minimum, maximum, median, and quartiles of the optimal cost for each instance group. The triangles in the box plots indicate the mean. Electronic companions EC.2 and EC.3 present the detailed results on the min-cost and min-energy graphs, respectively.

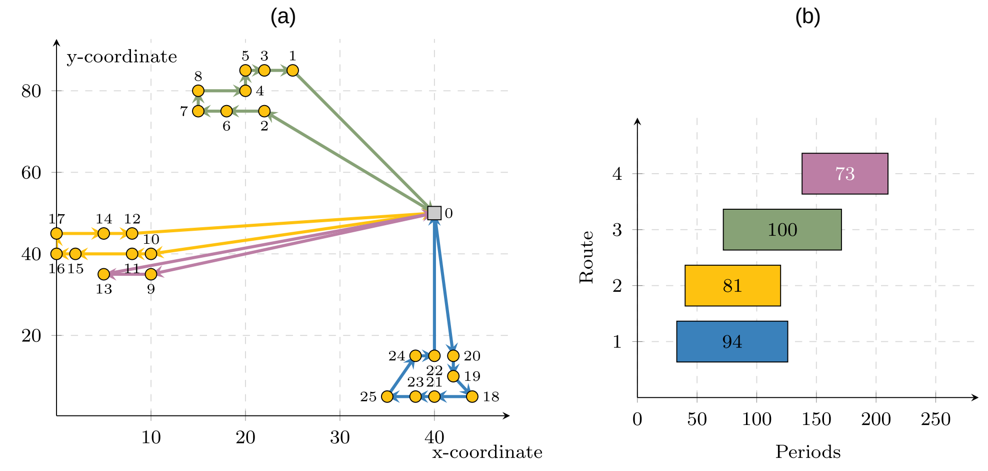
The plots clearly demonstrate that the multigraph representation consistently outperforms both 1-graph representations in terms of solution cost. To be more specific, on the multigraph instances, we find 112 better solutions with an average improvement of 13.0% compared with the min-energy instances. For the min-cost instances, these values are 76 and 2.9%, respectively. From a statistical standpoint, the multigraph instances show significant improvement over the min-energy ones, with the first, second, and third quartiles of these improvements being 11.0%, 12.3%, and 14.5%, respectively. The proximity between these quartiles indicates a concentrated

Figure 7. (Color online) RC104-25 Optimal Solution on the Multigraph



Notes. (a) Routing solution. (b) Charging schedule.

Figure 8. (Color online) RC104-25 Optimal Solution on the Min-Cost Graph



Notes. (a) Routing solution. (b) Charging schedule.

distribution of improvements, with most of them clustered around the median. With the min-cost graph instances, these quartiles stand at 0.0%, 1.3%, and 4.4%, suggesting that the distribution of improvements is skewed toward the right. Although the multigraph's improvements over the min-cost instances are generally modest, this skewness indicates higher benefits in some cases. However, these benefits are less pronounced overall than those observed against the min-energy instances. Electronic companion EC.1 presents the improvements of the multigraph over the min-energy and min-cost instances.

Although these results are unsurprising in the case of the min-energy graph, they may appear less intuitive regarding the min-cost graph. One contributing factor to this behavior is that solutions for the min-cost instances often use more EVs. In total, the min-cost graph instances require 7.3% more EVs than the multigraph instances. Because of the inability to balance energy consumption and travel distance, routes in the min-cost graph instances are constrained in the number of customers they can serve. Consequently, more routes are needed. To better illustrate how the BPC algorithm exploits the trade-offs offered by the multigraph representation, we present next the routes and the charging schedule in the optimal solution for the three versions of the same instance.

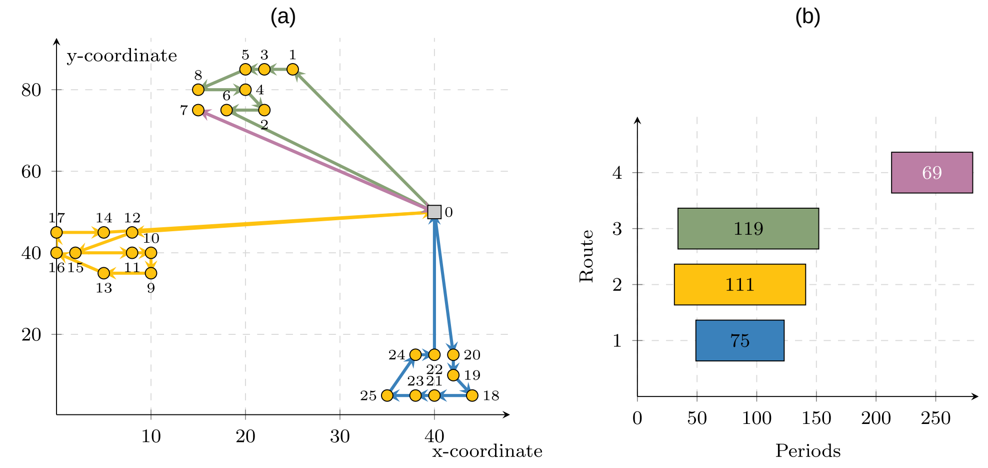
Figures 7–9 show the optimal solutions to instance RC104-25 on the multi-, min-cost, and min-energy graphs, respectively. Additionally, Table 3 presents the first and last charging periods, the amount of energy when leaving the depot, the cost, and the load of each route in these solutions. It also shows the total energy, cost, and load in bold. For this instance, $B = 3$, $\tilde{T} = 291$, $E = 1{,}600$, and $Q = 200$.

Route 1 (blue) follows the same sequence of customers in all three solutions. Because it does not use the exact same arcs, this route consumes less energy (1,347 dWh versus 1,522 dWh) on the min-energy graph, leading to a shorter charging operation (75 versus 94 periods), but it covers a shorter distance (101.7 km versus 108.0 km) on the multi- and min-cost graphs. Route 2 (yellow) visits the same set of customers on both the multigraph and min-energy graph solutions. However, the multigraph solution exploits the alternative arcs to do it more cost-effectively (115.3 km versus 119.8 km). It is also worth looking at the same set of customers in the min-cost solution. Because of their distance from the depot and the higher energy consumption associated with min-cost arcs, a single vehicle cannot serve the entire set of customers. Consequently, an additional route (route 4 in magenta) is needed to complete the task. Route 3 (green) visits the same sequence of customers (using the same arcs) in the

Table 3. Benefits of the Multigraph Representation on Instance RC104-25

	Multigraph					Min-cost graph					Min-energy graph				
	Charging periods					Charging periods					Charging periods				
Route	First	Last	Energy	Cost	Load	First	Last	Energy	Cost	Load	First	Last	Energy	Cost	Load
1	33	126	1,522	101.7	180	33	126	1,522	101.7	180	49	123	1,347	108.0	180
2	2	124	1,600	115.3	190	40	120	1,404	86.9	160	31	141	1,568	119.8	190
3	72	171	1,539	95.8	170	72	171	1,539	95.8	170	34	152	1,590	111.3	150
4	–	–	–	–	–	138	210	1,313	76.5	30	213	281	1,234	70.6	20
Total	–	–	**4,661**	**312.8**	**540**	–	–	**5,778**	**360.9**	**540**	–	–	**5,739**	**409.7**	**540**

Figure 9. (Color online) RC104-25 Optimal Solution on the Min-Energy Graph



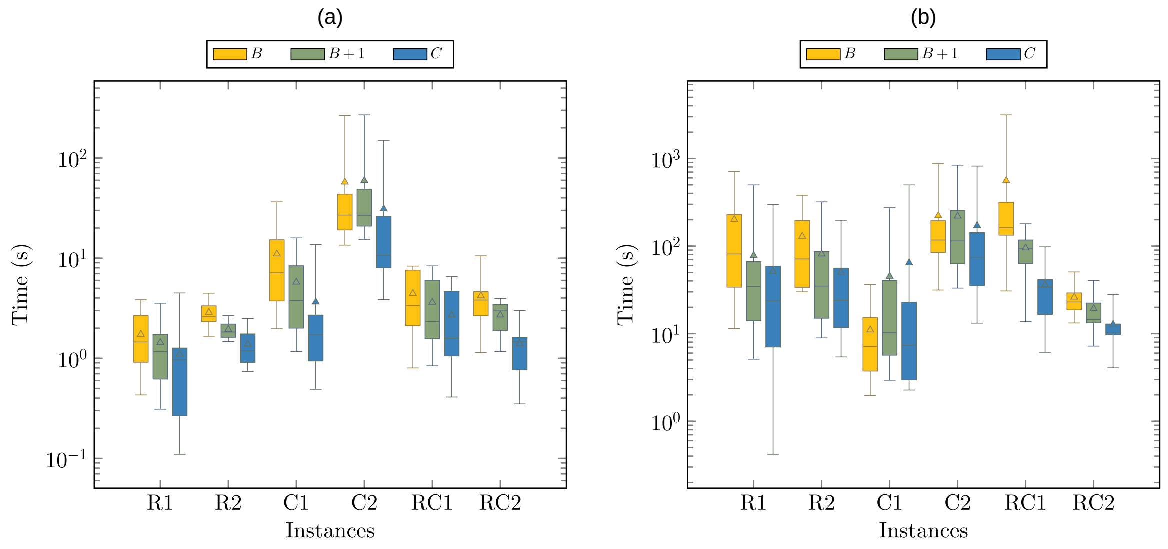
Notes. (a) Routing solution. (b) Charging schedule.

multigraph and min-cost solutions. Interestingly, that is not the case in the min-energy solution. Because min-energy arcs have a longer travel time, it is impossible for a single route to serve the same set of customers within their time windows. Therefore, the solution includes an additional single-customer route (route 4 in magenta). A closer examination of the charging schedule in Figure 9(b) highlights an interesting characteristic of our solutions. Indeed, the charging operation of route 4 starts several periods after the other routes have left the depot. This decision is clearly not driven by charging infrastructure limitations. Instead, it results from our backward labeling algorithm, which schedules charging operations to finish as late as possible, aiming to reduce calendar aging in the batteries of the EVs.

5.4. Impact of Charging Scheduling on Computational Time

We conducted an additional experiment to assess the impact of the number of chargers on the difficulty of solving the mE-VRSPTW. For each of the 112 instances, we created two additional versions: one with $B + 1$ chargers and one with C chargers (i.e., equal to the number of customers). Note that the latter relaxes the charging capacity constraints because the required number of vehicles cannot exceed C for any given instance. Figure 10 presents the box plots of the computational time for the resulting 336 (= 3 × 112) instances. Note that the y-axis is on a logarithmic scale.

Figure 10. (Color online) Impact of Charging Scheduling on Computational Time



Notes. (a) Twenty-five-customer instances. (b) Fifty-customer instances.

The plots show that the CPU time decreases as the charging capacity constraints become more relaxed. Instances with $B + 1$ chargers are solved around one-and-a-half times faster than those with B chargers. Furthermore, instances with fully relaxed charging capacity constraints (i.e., with C chargers) are solved almost three times faster than the original ones. These results highlight the complexity of dealing with the intertwined routing and scheduling decisions in the mE-VRSPTW. Detailed results for each instance with B, $B + 1$, and C chargers can be found in electronic companions EC.1, EC.4, and EC.5, respectively.

6. Conclusion

To facilitate this transition toward EVs in city logistics, we introduced the mE-VRSPTW that involves designing a set of routes for a fleet of EVs and planning their overnight charging operations at the depot before performing their routes. The objective is to minimize the total travel distance. The problem is defined on a multigraph, capturing the trade-off between energy consumption and travel distance.

To solve the mE-VRSPTW, we have designed a BPC algorithm that incorporates state-of-the-art techniques as well as problem-specific components. This BPC algorithm consistently and efficiently solved instances with up to 50 customers. Our experiments underscored the advantages of utilizing the multigraph representation in E-VRPs. Specifically, our findings demonstrate that optimal solutions achieved on multigraph instances outperform in terms of cost those obtained on the same instances defined on min-cost and min-energy 1-graphs. Furthermore, our experiments shed light on the complexity of solving the intertwined routing and charging scheduling problems. In particular, our results indicate that deactivating the scheduling component (i.e., having one charger per vehicle) can lead to an approximately threefold increase in solving speed for the same instance.

The central role of EVs in the urgent task of replacing fossil fuels with renewable energy sources is transforming the nature of routing and scheduling problems, exposing several fascinating research opportunities. Many research streams can spring from this work. Furthermore, known techniques like route enumeration and variable fixing by reduced cost could be used to enhance the algorithm and solve larger instances. From a practical standpoint, the problem could incorporate additional EV energy consumption characteristics like its dependency on the load onboard. Tackling the mE-VRSPTW over a multiday planning horizon or accounting for charging-related costs is also an interesting avenue.

Acknowledgments

The authors thank the anonymous referees for their valuable comments and Daniel S. Barahona for his clever observations during the early stages of this research.

Appendix A. Notation for Problem Statement and Formulation

Table A.1 presents the notation for problem statements and formulations.

Table A.1. Notation for Problem Statement and Formulation

Notation	Description
$\mathcal{C}$	Set of C customers.
$\mathcal{G} = (\mathcal{V}, \mathcal{A})$	Directed multigraph.
$\mathcal{V}$	Set of vertices.
$\mathcal{A}$	Multiset of arcs.
$\mathcal{A}_{(i,j)}$	Set of $K_{(i,j)}$ arcs from $i \in \mathcal{V}$ to $j \in \mathcal{V}$.
$Q \in \mathbb{Z}_+$	Load capacity of each EV.
$E \in \mathbb{Z}_+$	Maximum amount of energy that an EV can store.
$\underline{T}, \overline{T} \in \mathbb{Z}_+$	Depot's time window opening and closing.
$q_i \in \{0, 1 \ldots, Q\}$	Load associated with vertex $i \in \mathcal{V}$.
$\underline{t}_i, \overline{t}_i \in [\underline{T}, \overline{T}]$	Time window opening and closing of vertex $i \in \mathcal{V}$.
$c_{(i,j)^k} \geq 0$	Driving distance (or travel cost) along arc $(i,j)^k \in \mathcal{A}$.
$t_{(i,j)^k} \geq 0$	Travel time along arc $(i,j)^k \in \mathcal{A}$.
$e_{(i,j)^k} \in \{0, 1, \ldots, E\}$	Energy consumption along arc $(i,j)^k \in \mathcal{A}$.
$\underline{c}_{(i,j)}$, $\underline{t}_{(i,j)}$, $\underline{e}_{(i,j)}$	Minimum distance, travel time, and energy consumption from $i \in \mathcal{V}$ to $j \in \mathcal{V}$.
$B \in \mathbb{Z}_+$	Number of chargers at the depot.
$\mathcal{T}$	Set of periods at which vehicles can charge.
$\tilde{T} \in \mathbb{Z}_+$	Latest time at which a vehicle can depart from the depot.
f	Piecewise-linear recharging function.
$\mathcal{P}$	Set of P pieces of the recharging function.
$\phi_{p-1}, \phi_p \geq 0$	Start and end of the energy level associated with piece $p \in \mathcal{P}$.
$\theta_p > 0$	Recharging rate of piece $p \in \mathcal{P}$.

Table A.1. (Continued)

Notation	Description
f^{-1}	Inverse recharging function—refer to Equation (1).
Set-partitioning formulation (Section 3.2)	
$\mathcal{R}$	Set of all feasible routes.
$a_i^r \in \{0,1\}$	Parameter indicating whether route $r \in \mathcal{R}$ visits customer $i \in \mathcal{C}$.
$a_{(i,j)^k}^r \in \{0,1\}$	Parameter indicating whether route $r \in \mathcal{R}$ traverses arc $(i,j)^k \in \mathcal{A}$.
$b_t^r \in \{0,1\}$	Parameter indicating whether route $r \in \mathcal{R}$ charges during period $t \in \mathcal{T}$.
$c_r \in \mathbb{Z}_+$	Cost of route $r \in \mathcal{R}$—refer to Equation (2).
$\lambda_r \in \{0,1\}$	Variable indicating whether route $r \in \mathcal{R}$ is selected.

Appendix B. Notation for BPC Algorithm

Table B.1 presents the notation for the BPC algorithm.

Table B.1. Notation for BPC Algorithm

Notation	Description
The restricted master problem (RMP, Section 4.1)	
$\mathcal{R}' \subseteq \mathcal{R}$	Relatively small subset of routes.
$\alpha_i \in \mathbb{R}$	Dual variable associated with Constraint (4b) indexed by $i \in \mathcal{C}$.
$\beta_t \leq 0$	Dual variable associated with Constraint (4c) indexed by $t \in \mathcal{T}$.
The pricing problem (PP, Section 4.2)	
$\mathcal{G}_P = (\mathcal{V}_P, \mathcal{A}_P)$	Directed multigraph, where $\mathcal{V}_P = \{s\} \cup \mathcal{T} \cup \{\underline{0}\} \cup \mathcal{C} \cup \{\overline{0}\}$ and $\mathcal{A}_P = \mathcal{A} \cup \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$.
s	Dummy source vertex.
$\underline{0}, \overline{0}$	Source and sink depot vertices.
$\mathcal{A}$	Multiset of arcs connecting vertices of $\mathcal{V} = \{\underline{0}\} \cup \mathcal{C} \cup \{\overline{0}\}$.
$\mathcal{A}_1$	Set of arcs connecting charging period vertices with the source depot vertex.
$\mathcal{A}_2$	Set of arcs connecting successive charging period vertices.
$\mathcal{A}_3$	Set of arcs connecting the dummy source with charging period vertices.
$r_{(i,j)^k} \in \mathbb{R}$	Modified cost of arc $(i,j)^k \in \mathcal{A}_P$—refer to Equation (6).
$\mathcal{N}_i \subseteq \mathcal{C}$	Neighborhood containing the Δ closest customers to $i \in \mathcal{C}$ and customer i itself.
$\overline{\Delta} \in \mathbb{Z}_+$	Maximum neighborhood size.
$a_i^r \in \mathbb{Z}_+$	Number of times that route $r \in \mathcal{R}$ visits customer $i \in \mathcal{C}$.
$a_{(i,j)^k}^r \in \mathbb{Z}_+$	Number of times that route $r \in \mathcal{R}$ traverses arc $(i,j)^k \in \mathcal{A}$.
Labeling algorithm (Section 4.3)	
ℓ_i	Label associated with a subpath starting at vertex $i \in \mathcal{V}_P$.
$r(\ell_i) \in \mathbb{R}$	The cumulative reduced cost of label ℓ_i.
$q(\ell_i) \in \mathbb{Z}_+$	The remaining load capacity of label ℓ_i.
$t(\ell_i) \in [\underline{T}, \overline{T}]$	The latest time for starting service of label ℓ_i.
$e(\ell_i) \in \mathbb{Z}_+$	The remaining energy capacity of label ℓ_i.
$b(\ell_i) \in \mathbb{Z}_+$	The number of periods required to charge of label ℓ_i.
$\mathbb{I}_c^u(\ell_i) \in \{0,1\}$	Resource indicating whether customer $c \in \mathcal{C}$ is unreachable for label ℓ_i.
$\mathbb{I}_c^{ng}(\ell_i) \in \{0,1\}$	Resource indicating whether label ℓ_i visiting customer $c \in \mathcal{C}$ violates the cycling restrictions.
Valid inequalities (Section 4.4)	
$\pi \geq 0$	Dual variable associated with Constraint (9).
$\mathcal{S}$	Set of customer triplets for which SRCs have been separated.
$\sigma_S \leq 0$	Dual variable associated with Constraint (10) indexed by $S \in \mathcal{S}$.
$\overline{S} \in \mathbb{N}$	Maximum number of cuts that can be added simultaneously.
$\overline{C} \in \mathbb{N}$	Maximum number of times a customer may appear in a customer triplet.
$\epsilon > 0$	Violation threshold to separate cuts.
Branching rules (Section 4.5)	
$x_{(i,j)^k}, (i,j)^k \in \mathcal{A}_P$	Arc-flow variable.
$m_{(\underline{0},j)^k,(t,\underline{0})^1}$	Consecutive flow through arcs $(\underline{0},j)^k \in \mathcal{A}$ and $(t,\underline{0})^1 \in \mathcal{A}_1$.

Appendix C. Fifty-Customer RC Test Instances

Table C.1 presents 50-customer RC instances' minimum distance and energy consumption from the depot to the customers 26–34, using the setup described in Section 5.1. With an energy capacity of E = 1,600 dWh, arriving at most of these customers requires more than half of the EVs' driving range. As a result, some customers are unreachable or are serviced exclusively by one EV, which is impractical in real-life scenarios.

Table C.1. Fifty-Customer RC Instances' Minimum Distance and Energy Consumption from the Depot to Customers 26–34

Customer $j \in \mathcal{C}$	Min. distance $\underline{c}_{(0,j)}$ (km)	Min. energy consumption $\underline{e}_{(0,j)}$ (dWh)
26	58.5	933
27	57.0	873
28	55.7	898
29	52.2	811
30	52.0	848
31	50.2	786
32	51.0	836
33	51.4	875
34	47.4	749

To overcome this issue, we perform two changes in this group of instances. First, we set the EVs' energy capacity to $E = 20$ kWh, which is consistent with real-life practices as a long-range EV would be used for these long-distance trips. Second, we define the piecewise-linear recharging function by $\theta_1 = 13.49, \theta_2 = 7.14$, and $\theta_3 = 2.00$ kWh/h and $\phi_1 = 17.0, \phi_2 = 19.0$, and $\phi_3 = 20.0$ kWh. This means that the energy per period delivered by the chargers increases, but the time for a full recharge remains unchanged.

Appendix D. Computation of the Time in Use, Chargers in Use, and Time at Capacity

This appendix illustrates the computation of the time in use (TIU), chargers in use (CIU), and time at capacity (TAC) metrics on the optimal solution found for instance RC104-25. This solution is depicted in Figure 7 and its key elements (e.g., the first and last charging periods for each vehicle) are reported in Table 3 in the main body of the article. Recall there are $B = 3$ chargers at the depot. Let $\overline{\mathcal{R}} = \{r \in \mathcal{R} | \lambda_r = 1\}$ be the routes selected in the optimal solution and $n_t = \sum_{r \in \overline{\mathcal{R}}} b_t^r$ be the number of EVs charging at period $t \in \mathcal{T}$. Furthermore, let $\underline{b} = \min\{t \in \mathcal{T} | n_t \geq 1\}$ and $\overline{b} = \max\{t \in \mathcal{T} | n_t \geq 1\}$ be the first and last periods in which an EV is charging. As illustrated in Figure 7(b), $\underline{b} = 2$ and $\overline{b} = 171$ in the optimal solution for instance RC104-25. First, we compute the TIU as

$$\frac{\sum_{t=\underline{b}}^{\overline{b}} [n_t \geq 1]}{\overline{b} - \underline{b} + 1} = \frac{170}{171 - 2 + 1} = 1,$$

where the square brackets are Iverson brackets, that is, $[\mathbb{P}] = 1$ if $\mathbb{P}$ is true, and $[\mathbb{P}] = 0$ otherwise. Next, we compute the CIU as

$$\frac{\sum_{t=\underline{b}}^{\overline{b}} n_t}{B \cdot (\overline{b} - \underline{b} + 1)} = \frac{(1 \cdot 31) + (2 \cdot 39) + (3 \cdot 53) + (2 \cdot 2) + (1 \cdot 45)}{3 \cdot 170} = \frac{317}{510} = 0.6215,$$

considering that from $t = 2$ to $t = 32$ (31 periods), one EV is charging, from $t = 33$ to $t = 71$ (39 periods), two EVs are charging, and so forth. Finally, Equation (D.1) computes the TAC, noting that all three EVs charge simultaneously during 53 $(= 124 - 72 + 1)$ periods, and there are $B = 3$ chargers at the depot.

$$\frac{\sum_{t=\underline{b}}^{\overline{b}} [n_t = B]}{\overline{b} - \underline{b} + 1} = \frac{53}{170} = 0.3117. \tag{D.1}$$

References

Alvo M, Angulo G, Klapp MA (2021) An exact solution approach for an electric bus dispatch problem. *Transportation Res. Part E Logist. Transportation Rev.* 156:102528.

Asamer J, Graser A, Heilmann B, Ruthmair M (2016) Sensitivity analysis for energy demand estimation of electric vehicles. *Transportation Res. Part D Transport Environ.* 46:182–199.

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

Bektaş T, Ehmke JF, Psaraftis HN, Puchinger J (2019) The role of operational research in green freight transportation. *Eur. J. Oper. Res.* 274(3):807–823.

Ben Ticha H, 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.

Ben Ticha H, Absi N, Feillet D, Quilliot A (2018) Vehicle routing problems with road-network information: State of the art. *Networks* 72(3):393–406.

Bruglieri M, Mancini S, Pisacane O (2021) A more efficient cutting planes approach for the green vehicle routing problem with capacitated alternative fuel stations. *Optim. Lett.* 15(8):2813–2829.

Contardo C, Desaulniers G, Lessard F (2015) Reaching the elementary lower bound in the vehicle routing problem with time windows. *Networks* 65(1):88–99.

Costa L, Contardo C, Desaulniers G (2019) Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Sci.* 53(4):946–985.

Davis BA, Figliozzi MA (2013) A methodology to evaluate the competitiveness of electric delivery trucks. *Transportation Res. Part E Logist. Transportation Rev.* 49(1):8–23.

Desaulniers G (2010) Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Oper. Res.* 58(1):179–192.

Desaulniers G, Gschwind T, Irnich S (2020) Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models. *Transportation Sci.* 54(5):1170–1188.

Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized *k*-path inequalities for the vehicle routing problem with time windows. *Transportation Sci.* 42(3):387–404.

Desaulniers G, Pecin D, Contardo C (2019) Selective pricing in branch-price-and-cut algorithms for vehicle routing. *EURO J. Transportation Logist.* 8(2):147–168.

Desaulniers G, Errico F, Irnich S, Schneider M (2016) Exact algorithms for electric vehicle-routing problems with time windows. *Oper. Res.* 64(6):1388–1405.

Dror M (1994) Note on the complexity of the shortest path models for column generation in VRPTW. *Oper. Res.* 42(5):977–978.

Electric Vehicle Database (2022) Peugeot iOn. Accessed June 29, 2022, https://ev-database.org/car/1095/Peugeot-iOn.

Florio AM, Absi N, Feillet D (2021) Routing electric vehicles on congested street networks. *Transportation Sci.* 55(1):238–256.

Froger A, Jabali O, Mendoza JE, Laporte G (2022) The electric vehicle routing problem with capacitated charging stations. *Transportation Sci.* 56(2):460–482.

Froger A, Mendoza JE, Jabali O, Laporte G (2019) Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Comput. Oper. Res.* 104:256–294.

Gan L, Topcu U, Low SH (2012) Stochastic distributed protocol for electric vehicle charging with discrete charging rate. *2012 IEEE Power Energy Soc. General Meeting* (IEEE, Piscataway, NJ), 1–8.

Goeke D, Schneider M (2015) Routing a mixed fleet of electric and conventional vehicles. *Eur. J. Oper. Res.* 245(1):81–99.

Hiermann G, Puchinger J, Ropke S, Hartl RF (2016) The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *Eur. J. Oper. Res.* 252(3):995–1018.

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon MM, eds. *Column Generation* (Springer, Berlin), 33–65.

Jepsen M, Petersen B, Spoorendonk S, Pisinger D (2008) Subset-row inequalities applied to the vehicle-routing problem with time windows. *Oper. Res.* 56(2):497–511.

Kohl N, Desrosiers J, Madsen OB, Solomon MM, Soumis F (1999) 2-path cuts for the vehicle routing problem with time windows. *Transportation Sci.* 33(1):101–116.

Kullman ND, Goodson JC, Mendoza JE (2021a) Electric vehicle routing with public charging stations. *Transportation Sci.* 55(3):637–659.

Kullman ND, Froger A, Goodson JC, Mendoza JE (2021b) frvcpy: An open-source solver for the fixed route vehicle charging problem. *INFORMS J. Comput.* 33(4):1277–1283.

Lam E, Desaulniers G, Stuckey PJ (2022) Branch-and-cut-and-price for the electric vehicle routing problem with time windows, piecewise-linear recharging and capacitated recharging stations. *Comput. Oper. Res.* 145:105870.

Mendoza JE, Guéret C, Hoskins M, Lobit H, Pillac V, Vidal T, Vigo D (2014) VRP-REP: A vehicle routing community repository. *VeRoLog'14. Third Meeting EURO Working Group Vehicle Routing Logist. Optim.* (Association of European Operational Research Societies, Leeds, UK).

Michail D, Kinable J, Naveh B, Sichi JV (2020) JGraphT—A Java library for graph data structures and algorithms. *ACM Trans. Math. Software* 46(2):1–29.

Montoya A, Guéret C, Mendoza JE, Villegas JG (2017) The electric vehicle routing problem with nonlinear charging function. *Transportation Res. Part B Methodological* 103:87–110.

Morganti E, Browne M (2018) Technical and operational obstacles to the adoption of electric vans in France and the UK: An operator perspective. *Transport Policy* 63:90–97.

Pelletier S, Jabali O, Laporte G (2016) 50th anniversary invited article-goods distribution with electric vehicles: Review and research perspectives. *Transportation Sci.* 50(1):3–22.

Pelletier S, Jabali O, Laporte G (2018) Charge scheduling for electric freight vehicles. *Transportation Res. Part B Methodological* 115:246–269.

Perumal SS, Lusby RM, Larsen J (2022) Electric bus planning & scheduling: A review of related problems and methodologies. *Eur. J. Oper. Res.* 301(2):395–413.

Sassi O, Oulamara A (2014) Joint scheduling and optimal charging of electric vehicles problem. Murgante B, Misra S, Rocha AMAC, Torre C, Rocha JG, Falcão M, Taniar D, et al., eds. *Internat. Conf. Comput. Sci. Its Appl.* (Springer, Cham, Switzerland), 76–91.

Schiffer M, Walther G (2018) An adaptive large neighborhood search for the location-routing problem with intra-route facilities. *Transportation Sci.* 52(2):331–352.

Schneider M, Stenger A, Goeke D (2014) The electric vehicle-routing problem with time windows and recharging stations. *Transportation Sci.* 48(4):500–520.

Schneider M, Stenger A, Hof J (2015) An adaptive VNS algorithm for vehicle routing problems with intermediate stops. *OR Spectrum* 37(2):353–387.

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* 35(2):254–265.

Sun B, Huang Z, Tan X, Tsang DHK (2018) Optimal scheduling for electric vehicle charging with discrete charging levels in distribution grid. *IEEE Trans. Smart Grid* 9(2):624–634.

Vendé P, Desaulniers G, Kergosien Y, Mendoza JE (2023) Matheuristics for a multi-day electric bus assignment and overnight recharge scheduling problem. *Transportation Res. Part C Emerging Tech.* 156:104360.

Yamín D, Desaulniers G, Mendoza JE (2024) The electric vehicle routing and overnight charging scheduling problem on a multigraph. http://dx.doi.org/10.1287/ijoc.2023.0404.cd, https://github.com/INFORMSJoC/2023.0404.

Zhou Y, Meng Q, Ong GP (2022) Electric bus charging scheduling for a single public transport route considering nonlinear charging profile and battery degradation effect. *Transportation Res. Part B Methodological* 159:49–75.