Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Branch-Price-and-Cut for the Electric Vehicle Routing Problem with Heterogeneous Recharging Technologies and Nonlinear Recharging Functions

Gaute Messel Nafstad ; , Guy Desaulniers , Magnus Stålhane ;

To cite this article:
Gaute Messel Nafstad ; , Guy Desaulniers , Magnus Stålhane ; (2025) Branch-Price-and-Cut for the Electric Vehicle Routing Problem with Heterogeneous Recharging Technologies and Nonlinear Recharging Functions. Transportation Science 59(3):628-646. https://doi.org/10.1287/trsc.2024.0725

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

TRANSPORTATION SCIENCE
Vol. 59, No. 3, May–June 2025, pp. 628–646
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Branch-Price-and-Cut for the Electric Vehicle Routing Problem with Heterogeneous Recharging Technologies and Nonlinear Recharging Functions

Gaute Messel Nafstad,[a] Guy Desaulniers,[b,c] Magnus Stålhane[a,*]

[a] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, 7491 Trondheim, Norway; [b] Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Montreal, Quebec H3T 1J4, Canada; [c] Group for Research in Decision Analysis, HEC Montréal, Montreal, Quebec H3T 2A7, Canada

*Corresponding author

Contact: gaute.m.nafstad@ntnu.no (GMN); guy.desaulniers@gerad.ca, https://orcid.org/0000-0003-4469-9813 (GD); magnus.staalhane@ntnu.no, https://orcid.org/0009-0001-6441-9877 (MS)

Received: May 26, 2024
Revised: December 21, 2024
Accepted: February 6, 2025
Published Online in Articles in Advance: April 14, 2025

https://doi.org/10.1287/trsc.2024.0725

Abstract. As electric vehicles become increasingly prevalent, effective planning of their use becomes paramount. The electric vehicle routing problem, characterized by limited driving range and the need for recharging, poses unique challenges compared with traditional vehicle routing problems. This paper proposes a branch-price-and-cut solution method tailored for the electric vehicle routing problem with time windows, heterogeneous recharging technologies, and nonlinear charging functions (E-VRPTW-NL). The methodology differs from previous methods proposed in the literature by handling nonlinear recharging functions in the pricing problem. The pricing problem is solved by a bidirectional labeling algorithm that efficiently handles the complex interdependency between time and state of charge during recharge scheduling. The proposed solution method is tested on both benchmark instances from the literature as well as new instances. Tests show that the solution method is competitive with well-known solution methods from the literature on simpler variants of the problem. The computational results also indicate that the proposed method can solve new E-VRPTW-NL instances with up to 100 customers and 21 recharge locations within one hour. Further analysis explores how simplifying the modeling of the recharging process affects solution feasibility and cost. The results show that keeping the heterogeneity of the recharging functions is crucial, whereas simplifying the shape of each recharging function has limited impact.

Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2024.0725.

Keywords: electric vehicle routing • branch-price-and-cut

1. Introduction

The world's fleet of electric vehicles is growing. Important actors, such as the European Union, have indicated that they will start to phase out petrol vehicles in order to reduce greenhouse gas emissions, with several countries signing an agreement to stop sales of new emitting cars by 2040 during the 2021 United Nations Climate Change Conference (Carey and Steitz 2021). Today, most electric vehicles store energy in batteries. Battery capacity has evolved significantly in the last decade (Mohammadi and Saif 2023), mitigating some of the early limitations of electric vehicle range. For instance, in daily delivery operations within metropolitan areas, the advancement of battery technologies has reduced the need for en route charging. However, en route charging remains necessary for intercity operations and in operations requiring extended or continuous (round-the-clock) vehicle use or lightweight vehicles, like drones (Vidal, Laporte, and Matl 2020, Su et al. 2024, Adsanver, Coban, and Balcik 2025). Therefore, route planning that takes into account limited driving range and the possibility of en route recharging is still an important research topic.

Planning routes for a fleet of petrol vehicles is usually referred to as the vehicle routing problem (VRP). Petrol vehicles also have a limited driving range. However, because the time needed to refuel them is negligible and the availability of gas stations is ubiquitous, refueling is usually excluded from route planning. The VRP is defined as the problem of routing a fleet of homogeneous

vehicles to visit a set of customers, where each customer must be visited once to receive a delivery of a specified amount of product, which is referred to as its demand. Each vehicle starts at a central depot, visits a subset of customers with a total demand less than the vehicle capacity, and ends its route back at the depot. The objective is to minimize the total distance traveled by the vehicles. One important extension of the VRP is the vehicle routing problem with time windows (VRPTW), where a time interval called a time window is associated with each customer that restricts when the customer may be served.

The electric vehicle routing problem (E-VRP) extends the VRP by considering electric vehicles with a limited battery capacity, which restricts how far they can travel before having to either recharge or end their route. The vehicles may visit dedicated recharging stations in addition to the depot and the customer locations to recharge their batteries. The recharging technologies available at recharging stations are represented by *recharging functions*, which describe the relationship between charging time and the increase in the battery power level. This power level is often referred to as the state of charge (SoC). The objective of the E-VRP is usually to minimize either the total distance traveled or the total time spent on the routes, including travel time and recharging time.

In the literature, recharging functions have been modeled in various ways, which leads to different versions of the E-VRP. The way that the recharging is modeled varies in two important aspects, the first being whether recharging stations have the same or different recharging technologies. The other is whether the recharging functions are assumed to be linear or nonlinear. Having both heterogeneous recharging technologies and nonlinear charging makes the problem considerably more complex. In this paper, we develop a new algorithm, namely a branch-price-and-cut (BP&C) algorithm, for solving the electric vehicle routing problem with time windows, heterogeneous recharging technologies, and nonlinear recharging functions (E-VRPTW-NL).

1.1. Literature Review

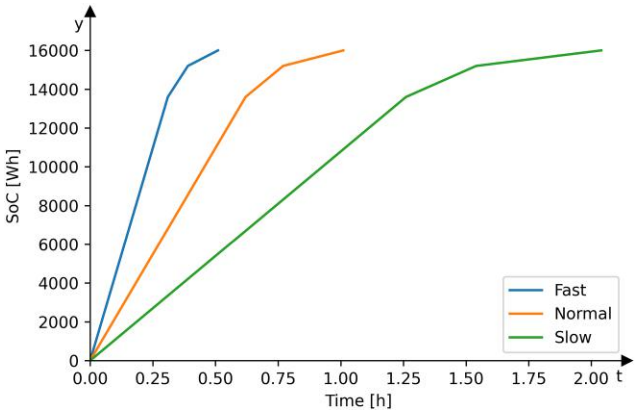
The E-VRP has received increasing attention in the last decade. The green vehicle routing problem, introduced by Erdoğan and Miller-Hooks (2012), was among the first to consider dedicated recharging locations in the VRP. In subsequent years, many extensions of the problem, with different attributes, have appeared in the E-VRP literature. A survey by Erdelić and Carić (2019) presents many of these extensions and the tailored exact and heuristic solution methods used. Many of these variations relate to the modeling of the recharging procedure.

Early works (Erdoğan and Miller-Hooks 2012, Omidvar and Tavakkoli-Moghaddam 2012, Preis, Frank, and Nachtigall 2014) considered a constant-time recharging process, similar to the process of battery swapping. Later, other recharging modeling alternatives have been developed to better mimic the real recharging possibilities. One of these possibilities is to allow for partial recharging. The electric vehicle routing problem with partial recharging using a linear recharging function (E-VRP-L) and the electric vehicle routing problem with partial recharging and time windows using a linear recharging function (E-VRPTW-L) were introduced by Schneider, Stenger, and Goeke (2014), who design a heuristic for the problem. This feature adds flexibility to the planning, and hence, it might give improved solutions. Desaulniers et al. (2016) and Duman, Taş, and Çatay (2022) develop exact BP&C algorithms for the E-VRPTW-L, allowing for multiple recharges per route and improved versions for the special case of at most one recharge per route. The former also presents a BP&C method for the case when only fully recharging the battery is allowed.

The E-VRP-L with multiple recharging technologies, where each recharging station offers several types of recharging technologies at varying costs, was introduced by Felipe et al. (2014), who developed a heuristic to solve the problem. Later, exact BP&C algorithms were proposed by Ceselli et al. (2021) and Bezzi, Ceselli, and Righini (2023). The former relies on a formulation where the columns generated represent *no charge segments*. Such a segment is a partial path, where the origin and destination are either the depot or a recharging station, and the remaining visits on the partial path are to customer locations. The latter generates columns representing full paths originating and ending at the depot using a bidirectional labeling algorithm. Comparing the results shows that using full paths leads to a stronger formulation, with better dual bounds, and also, leads to lower computational times.

In reality, the recharging function is a nonlinear concave function with regard to time (Pelletier et al. 2017). According to Montoya et al. (2017), the recharging function is known to be close to linear on the interval from 0% SoC to around 80%, and then, the recharging rate decreases exponentially for the remaining 20%. They further argue that a piecewise linear function with three segments, as depicted in Figure 1, is a sufficiently good approximation of the recharging function. Based on this observation, they introduce the electric vehicle routing problem with nonlinear charging functions (E-VRP-NL) and heterogeneous recharging technologies. They further present a set of benchmark instances for the problem, which they solve using both a mixed-integer linear programming (MILP) formulation and a hybrid metaheuristic that combines iterated local search with heuristic concentration. Two tighter MILP formulations, one arc based and the other path based, are devised by Froger et al. (2019), whereas a three-index formulation and

Figure 1. (Color online) The Recharging Functions Used in the Instances of Montoya et al. (2017)



an adaptive large neighborhood search are presented in Kancharla and Ramadurai (2020). An exact branch-and-price (B&P) method for the single-vehicle E-VRP-NL with strictly concave recharging functions is designed by Lee (2021). He uses a method similar to that of Ceselli et al. (2021) by generating no-charge segments in the pricing problem and handling the nonlinear recharging functions in the master problem.

Lam, Desaulniers, and Stuckey (2022) are the first to address a variant of the E-VRPTW-NL. They do not include heterogeneous recharging technologies; however, they do take into account a limit on the number of vehicles that can simultaneously charge at a recharging station. The problem is solved using a combination of BP&C and logic-based Benders cuts. Recently and in parallel with this work, a BP&C method for the time-dependent E-VRPTW-NL was developed by Lera-Romero, Miranda Bront, and Soulignac (2024). The pricing problem is solved by a monodirectional labeling algorithm, where the interdependency between time and SoC is handled by storing a piecewise linear function in every label. They further show how the waiting time at the charging stations and time-dependent travel times can be stored in the same function.

Baum et al. (2019, 2020) present works on problems closely related to the one appearing as the pricing problem when solving the E-VRPTW-NL by B&P. Baum et al. (2019) solve a shortest-path problem, where the network contains recharging nodes with heterogeneous recharging technologies and where the recharging functions are modeled by piecewise linear functions. To solve this problem, they devise a labeling algorithm, where labels are of constant size. The downside of their approach is that a single extension of a label might cause multiple new labels rather than one. Baum et al. (2020) present a similar shortest-path problem with battery limitations, where the trade-off between time and energy stems from the modeling of energy consumption as a function of speed rather than nonlinear charging functions. The problem is solved by a tailored labeling algorithm. Cassia et al. (2022) extend the work of Baum et al. (2019) by including prize collection at recharging stations. They model the problem as an MILP and propose an A* search heuristic to solve it.

1.2. Contributions

The main contribution of this paper is the development of an exact BP&C method to solve the E-VRPTW-NL, where the novelty lies in the bidirectional labeling algorithm used to solve the pricing problem. In existing bidirectional algorithms for the E-VRPTW-L, such as that proposed by Desaulniers et al. (2016), the recharging function is linear, and the labeling algorithm, therefore, only needs to keep track of, and update, the minimal and maximal SoC along a partial path to represent the entire SoC function. For the E-VRPTW-NL, the pricing problem is significantly more challenging to solve because we need to explicitly calculate and store the entire SoC function whenever a label is extended. To accomplish this, we model the trade-off between time and SoC as a recursive function and propose an algorithm for computing the complete charging function when extending a partial path in each of the forward and backward directions. The time complexity of the algorithm is contingent upon the recharging functions considered. Additionally, we propose a linear time algorithm for the special case of piecewise linear recharging functions. The method proposed for managing the nonlinear trade-off between time and SoC can also be used for assessing cost and route feasibility for a fixed route, something that is often a crucial part of many heuristics for E-VRPs (Froger et al. 2019).

To test the proposed algorithm, we introduce a new set of benchmark instances obtained by extending existing instances with linear recharging functions. In addition, we test our BP&C algorithm on existing E-VRPTW-L and E-VRP-NL benchmark instances. The results show that our method is competitive with tailored solution methods for the E-VRPTW-L and outperforms existing methods for the E-VRP-NL. Furthermore, we show that using bidirectional labeling for solving the pricing problem exactly is significantly faster than using a monodirectional labeling algorithm for the E-VRPTW-NL.

Finally, we use the instances generated to analyze the effect of various simplifications of the recharging functions. This analysis gives insight into which simplifications are preferable in different problem settings. The key takeaways are that using different recharging functions is more important than how each recharging function is modeled and that modeling the recharging functions correctly is less important when the customers have relatively wide time windows.

The remainder of the paper is organized as follows. A formal problem description and notation are provided in Section 2. Sections 3 and 4 describe the proposed BP&C algorithm, where Section 3 focuses on a fixed path and Section 4 focuses on the complete algorithm. Numerical results are reported in Section 5 before providing concluding remarks in Section 6.

2. Problem Statement and Mathematical Formulation

The E-VRPTW-NL can be defined on a directed graph $G = (N, A)$, where N is the set of nodes and $A \subseteq \{(i,j) \mid i,j \in N, i \neq j)\}$ is the set of arcs. Set N consists of the mutually exclusive sets C, S and $\{d(s), d(e)\}$. Set C is the set of customer nodes, S is the set of recharging nodes, and $d(s)$ and $d(e)$ are the source and sink nodes of the graph, both representing the depot. For each node $i \in N$, let q_i denote its demand and $[\underline{T}_i, \overline{T}_i]$ denote its time window within which the service must start. Note that $q_i = 0$ for all $i \in N \setminus C$. Furthermore, let $t_{i,j}$, $e_{i,j}$, and $c_{i,j}$ represent the travel time, energy consumption, and distance associated with arc $(i,j) \in A$. For ease of notation, any service time at node $i \in C$ is included in $t_{i,j}$.

Each recharging node is associated with a concave and continuous recharging function $h_i(t)$ (not necessarily differentiable), giving the SoC reached after having recharged an initially depleted battery for t time units. Examples of $h_i(t)$ functions are given in Figure 1. Let $g_i(y_1, y_2)$ denote the amount of time needed to recharge from an SoC of y_1 to an SoC of y_2 at node i. If i is a recharging node, it can be expressed as $g_i(y_1, y_2) = h_i^{-1}(y_2) - h_i^{-1}(y_1)$. If i is a customer node, then this function is given by

$$g_i(y_1, y_2) = \begin{cases} 0 & \text{if } y_1 = y_2 \\ \infty & \text{otherwise.} \end{cases} \tag{1}$$

We consider an unlimited fleet of homogeneous vehicles, where each vehicle has a battery capacity B and a load capacity Q. A route can be modeled as a path through G complying with the following route restrictions. (1) The path must start and end at the depot, (2) the total demand of all visited customers must not exceed Q, (3) there must exist a feasible schedule for the path where the service at each visited customer node starts within its time window, and (4) the battery SoC stays in the interval $[0, B]$ throughout the path. The objective is to find a set of feasible routes such that every customer is serviced once and the total cost of these routes is minimized.

Let Ω, indexed by p, be the set of all feasible routes. Let $(i_0, \ldots, i_n)$ be a sequence of nodes representing such a feasible route, where i_0 and i_n represent the depot and the intermediate nodes are customer or recharging nodes. The cost c_p of route p is computed as $c_p = \sum_{k=0}^{n-1} c_{i_k, i_{k+1}}$ and might represent, for instance, the distance traveled along the route. The number of times that customer node i is visited on path p is denoted a_{ip}. Let λ_p represent a binary variable stating whether path p is part of the solution or not.

With this notation, the E-VRPTW-NL can be modeled as follows:

$$\min \sum_{p \in \Omega} c_p \lambda_p, \tag{2}$$

$$\text{s.t.} \sum_{p \in \Omega} a_{ip} \lambda_p = 1, \quad \forall i \in C, \tag{3}$$

$$\lambda_p \in \{0,1\}, \quad \forall p \in \Omega. \tag{4}$$

The total route cost is minimized in the objective function (2). A single visit to every customer is ensured with Constraints (3), whereas Constraints (4) enforce binary values on the route variables. Note that the parameters a_{ip} are typically binary, but they can take a value larger than one when considering the *ng*-path relaxation discussed in Section 4.1.4.

3. Optimal Charging Schedule for a Given Partial Path

For a given partial path, there are, in general, infinitely many feasible values of SoC at any node along the path because the SoC is a function of the time spent charging, which is a continuous variable. Hence, the connection between the SoC and the required time leads to a trade-off between the two. In the following sections, this trade-off is described through forward and backward recursions, which are required in the solution method presented in Section 4.

3.1. Forward Recursion

Let $(i_0, i_1, \ldots, i_n)$ be a partial path beginning at $i_0 = d(s)$. For each node i_k, $k = 0, \ldots, n$, let $f_k(y)$ be a function providing the minimum departure time from i_k if the vehicle leaves with an SoC $y \in [0, B]$. This function is defined by the following forward recursion:

$$f_0(y) = 0, \qquad \forall y \in [0, B], \tag{5}$$

$$f_k(y) = \min_{y_0 \in [0,y]} \{\mathrm{TW}^{\mathrm{F}}_{i_k}(f_{k-1}(y_0 + e_{i_{k-1}, i_k}) + t_{i_{k-1}, i_k}) + g_{i_k}(y_0, y)\}, \quad \forall y \in [0, B], \quad \forall k \in \{1, \ldots, n\}. \tag{6}$$

In (6), y_0 represents the SoC when arriving at node i_k, and the function $TW_i^F(t)$ for $i = i_k$ encodes the adjustments needed to meet the time window at node i as follows:

$$\mathrm{TW}^{\mathrm{F}}_i(t) = \begin{cases} \underline{T}_i & \text{if } t < \underline{T}_i, \\ t, & \text{if } t \in [\underline{T}_i, \overline{T}_i], \\ \infty, & \text{if } t > \overline{T}_i. \end{cases} \tag{7}$$

Finally, we assume that $f_{k-1}(y) = \infty$ whenever $y > B$.

For a given partial path, the domain of $f_k(y)$, denoted $domain(f_k)$, is given by $\{y \in [0,B] \,|\, f_k(y) \neq \infty\}$ and represents all feasible SoC values when departing node i_k. An example of $f_k(y)$ is illustrated in Figure 2. One key observation from Figure 2 is that $f_k(y)$ may be constant for low values of y (i.e., different SoC values yield the same earliest departure time). This occurs when the SoC can be increased along the route without incurring a delay when departing from i_k. This might be possible in two situations. (i) The total energy consumed along the route does not exceed B, and thus, the initial SoC can be increased. (ii) After visiting a recharging node, the vehicle needs to wait at a customer node before the opening of its time window, giving the opportunity to charge more at the previous recharging event. Another key observation from Figure 2 is that even though all inverse recharging functions $h_i^{-1}(y)$ are convex, the resulting function $f_k(y)$ is not necessarily convex. Because $domain(f_k)$ is a continuous interval, it is not viable to directly calculate (6) for all feasible values of y in order to complete one step of the recursion. The following paragraphs describe how to calculate $f_k(y)$ for any feasible SoC value y.

For ease of exposition, we introduce the function $f_k^{\mathrm{TW}}(y) = \mathrm{TW}_{i_k}^{\mathrm{F}}(f_{k-1}(y + e_{i_{k-1},i_k}) + t_{i_{k-1},i_k})$ describing the minimum time at which charging at node i_k can start given that the vehicle arrives with an SoC of y. This function can be seen as shifting $f_{k-1}(y)$ as time increases and available energy decreases when traveling from i_{k-1} to i_k (see Figure 3). Function $f_k^{\mathrm{TW}}(y)$ adjusts the shifted function to stay within the time window of node i_k (see Figure 4). Introducing $f_k^{\mathrm{TW}}(y)$ enables the following rewriting of the recursion (6), which we henceforth use to explain the algorithm for calculating $f_k(y)$:

$$f_k(y) = \min_{y_0 \in [0,y]} \{f_k^{\mathrm{TW}}(y_0) + g_{i_k}(y_0, y)\}, \quad \forall y \in [0,B], \quad \forall k \in \{1,\dots,n\} \tag{8}$$

Figure 2. (Color online) Example of a Function $f_k(y)$

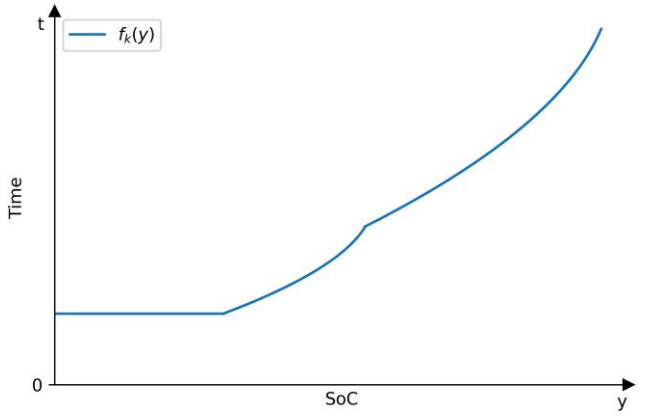
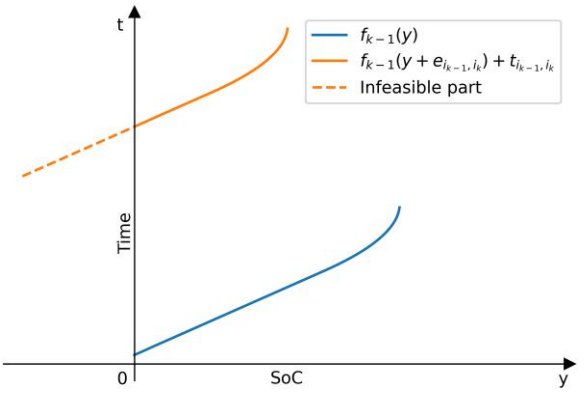


Figure 3. (Color online) Illustration of the Shifting Procedure

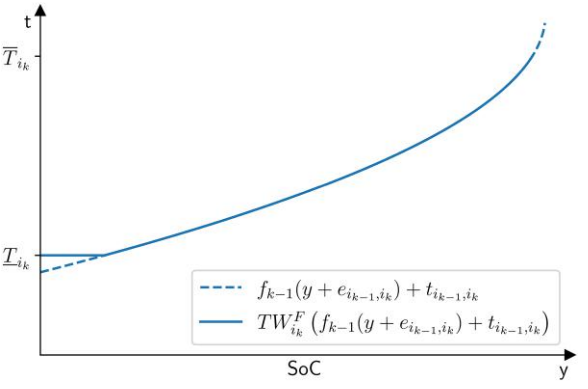


or equivalently,

$$f_k(y) = \begin{cases} \min\limits_{y_0 \in [0,y]} \{f_k^{\mathrm{TW}}(y_0) + h_{i_k}^{-1}(y) - h_{i_k}^{-1}(y_0)\}, & \text{if } i_k \in S, \\ f_k^{\mathrm{TW}}(y), & \text{otherwise,} \end{cases} \quad \forall y \in [0,B], \quad \forall k \in \{1,\dots,n\}. \tag{9}$$

Finding $f_k(y)$ for customer nodes is trivial, but for recharging nodes, it is more complicated. From (9), we identify two possible strategies for departing node $i_k \in S$ as early as possible with an SoC y. Either the vehicle arrives with an SoC $y_0 < y$ and charges the battery from an SoC y_0 to y, making $f_k(y) = f_k^{\mathrm{TW}}(y_0) + h_{i_k}^{-1}(y) - h_{i_k}^{-1}(y_0)$, or it arrives with $y_0 = y$, making $f_k(y) = f_k^{\mathrm{TW}}(y)$. We refer to these policies as *Charge* and *No charge*, respectively. In the following, we present Theorem 1, which given an SoC y_0, determines the optimal policy at y_0 and the maximal interval $[y_0, y_1]$ in which this policy does not change. In this theorem, we denote by $\frac{df}{dy^+}$ the right-hand derivative of a function f. For ease of exposition, let $f_k^{\mathrm{TW}}(y) = \infty$ and $\frac{df_k^{\mathrm{TW}}(y)}{dy^+} = \infty$ if $y \notin domain(f_k^{\mathrm{TW}})$.

Figure 4. (Color online) Adjusting to the Time Window $[\underline{T}_{i_k}, \overline{T}_{i_k}]$



Theorem 1. *Let $i_k \in S$ be a recharging node and $y_0 \in [0, B)$ be an SoC value. If $f_k(y_0) = f_k^{\mathrm{TW}}(y_0)$ and $\frac{df_k^{\mathrm{TW}}(y_0)}{dy^+} \leq \frac{dh_{i_k}^{-1}(y_0)}{dy^+}$, then $f_k(y) = f_k^{\mathrm{TW}}(y)$ for all $y \in [y_0, y_1]$, where $y_1 = \min\left\{B, \inf\left\{y' \mid y' > y_0, \frac{df_k^{\mathrm{TW}}(y')}{dy^+} > \frac{dh_{i_k}^{-1}(y')}{dy^+}\right\}\right\}$. Otherwise, $f_k(y) = f_k(y_0) + h_{i_k}^{-1}(y) - h_{i_k}^{-1}(y_0)$ for all $y \in [y_0, y_1]$, where $y_1 = \min\left\{B, \min\left\{y' \mid y' > y_0, f_k(y_0) + h_{i_k}^{-1}(y') - h_{i_k}^{-1}(y_0) = f_k^{\mathrm{TW}}(y'), \frac{df_k^{\mathrm{TW}}(y')}{dy^+} \leq \frac{dh_{i_k}^{-1}(y')}{dy^+}\right\}\right\}$.*

A proof of Theorem 1 is given in Online Appendix A. Note that this proof does not rely on the concavity of the function $h_i(t)$. Therefore, Theorem 1 may be applicable to more general charging functions. Figure 5 shows examples of $f_k^{\mathrm{TW}}(y)$, $h_{i_k}^{-1}$ and the resulting $f_k(y)$, where the functions are piecewise linear for illustrative purposes. Note that $h_{i_k}^{-1}$ has been positioned to coincide with f_k^{TW} at $y = 0$.

Based on Theorem 1, Algorithm 1 gradually finds all of the function pieces that constitute f_k by identifying a sequence of alternating SoC intervals where it is optimal to charge (*Charge*) and not to charge (*No charge*), respectively. The time complexity of Algorithm 1 is contingent upon the recharging functions considered and aligns with that of the algorithm employed to solve the minimization problems on lines 10 and 14 in Algorithm 1. Note that it is possible to simplify the calculation of y_1 on line 14 in Algorithm 1 to be $y_1 = \min\{B, \min\{y' \mid y' > y_0, f_k(y_0) + h_{i_k}^{-1}(y') - h_{i_k}^{-1}(y_0) = f_k^{\mathrm{TW}}(y')\}\}$ because this expression is a relaxation of the one above and therefore, always gives a smaller or equal interval $[y_0, y_1]$. However, this simplification may lead to additional iterations in the while loop because the next iteration may give $y_0 = y_1$ on line 10 in Algorithm 1 if $\frac{df_k^{\mathrm{TW}}(y_0)}{dy^+} > \frac{dh_{i_k}^{-1}(y_0)}{dy^+}$, thus only changing the policy back to *Charge*. Whether this change in the algorithm is computationally beneficial depends on the shape of the functions involved. Online Appendix B provides a lower-level pseudocode for the special case with piecewise linear h_i functions. In this case, the time complexity is linear with respect to the number of line segments in h_{i_k} and f_k^{TW}.

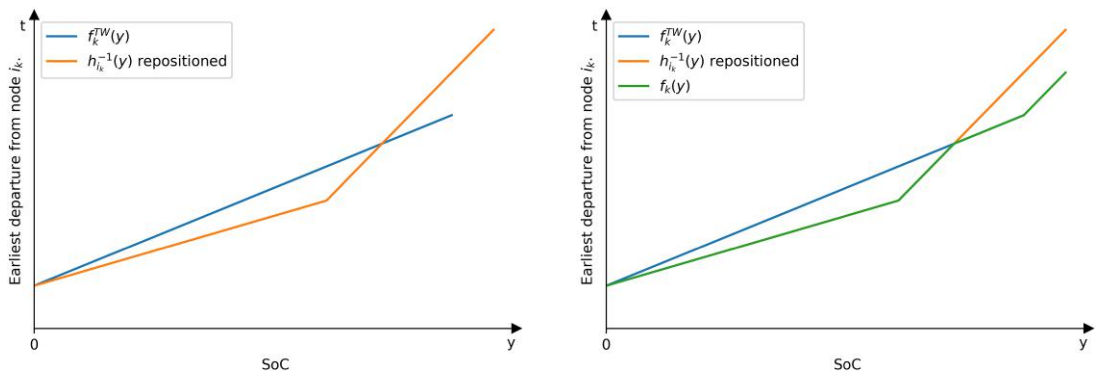
Algorithm 1 (Compute f_k)

1: **function** CHARGE(f_k^{TW}, h_{i_k})
2: $y_0 = 0$
3: State ← No charge
4: $f_k(y_0) = f_k^{\mathrm{TW}}(y_0)$
5: **if** $\frac{df_k^{\mathrm{TW}}(y_0)}{dy^+} > \frac{dh_{i_k}^{-1}(y_0)}{dy^+}$ **then**
6: State ← Charge
7: **end if**
8: **while** $y_0 < B$ **do**
9: **if** State = No charge **then**
10: $y_1 = \min\left\{B, \inf\left\{y' \mid y' > y_0, \frac{df_k^{\mathrm{TW}}(y')}{dy^+} > \frac{dh_{i_k}^{-1}(y')}{dy^+}\right\}\right\}$
11: $\forall y \in [y_0, y_1] : f_k(y) = f_k^{\mathrm{TW}}(y)$
12: State ← Charge
13: **else**
14: $y_1 = \min\left\{B, \min\left\{y' \mid y' > y_0, f_k(y_0) + h_{i_k}^{-1}(y') - h_{i_k}^{-1}(y_0) = f_k^{\mathrm{TW}}(y'), \frac{df_k^{\mathrm{TW}}(y')}{dy^+} \leq \frac{dh_{i_k}^{-1}(y')}{dy^+}\right\}\right\}$
15: $\forall y \in [y_0, y_1] : f_k(y) = f_k(y_0) + h_{i_k}^{-1}(y) - h_{i_k}^{-1}(y_0)$
16: State ← No charge
17: **end if**
18: $y_0 \leftarrow y_1$
19: **end while**
20: **end function**

3.2. Backward Recursion

Before presenting the details of the backward recursive function $w_k(y)$, we highlight a key difference to the forward recursive function. Because it is used in a labeling algorithm in Section 4, we define $w_k(y)$ to be the latest departure time from node i_k to fit with $f_k(y)$, which is defined as the earliest departure time from node i_k for an SoC of y. Function $f_k(y)$ involves three steps: travel, time window adjustment, and charging. The backward recursion used to obtain $w_k(y)$ needs to perform the same three steps but in the opposite sequence. This is illustrated in Figure 6, which shows that when calculating $w_k(y)$, we must first recharge at node i_{k+1} and then adjust for the time window before finally accounting for the travel between nodes i_k and i_{k+1}. With this in mind,

Figure 5. (Color online) An Example of $f_k^{\mathrm{TW}}(y)$, $h_{i_k}^{-1}$ and the Resulting $f_k(y)$



we present how to calculate the backward recursive function $w_k(y)$.

Let $(i_0, i_1, \ldots, i_n)$ be a partial path ending at $i_n = d(e)$. For each node i_k, $k = 0, \ldots, n$, let $w_k(y)$ be a function returning the latest departure time from i_k if the vehicle leaves with an SoC $y \in [0, B]$ (assuming a maximal arrival time at node i_n of $\overline{T}_n$). This function is defined by the following backward recursion:

$$w_n(y) = \overline{T}_{i_n}, \quad \forall y \in [0, B], \tag{10}$$

$$w_k(y) = \mathrm{TW}^B_{i_{k+1}}\left(\max_{y_1 \in [y - e_{i_k, i_{k+1}}, B]} \{w_{k+1}(y_1) - g_{i_{k+1}}(y - e_{i_k, i_{k+1}}, y_1)\}\right) - t_{i_k, i_{k+1}}, \ \forall y \in [0, B], \ \forall k \in \{0, \ldots, n-1\}. \tag{11}$$

In (11), y_1 represents the SoC at the departure from node i_{k+1}, and the function $\mathrm{TW}^B_i(t)$ for $i = i_{k+1}$ encodes the adjustments needed to meet the time window at node i as follows:

$$TW^B_i(t) = \begin{cases} -\infty, & \text{if } t < \underline{T}_i, \\ t, & \text{if } t \in [\underline{T}_i, \overline{T}_i], \\ \overline{T}_i, & \text{if } t > \overline{T}_i. \end{cases} \tag{12}$$

Finally, we assume that $w_{k+1}(y) = -\infty$ whenever $y < 0$.

The remainder of this subsection describes how to calculate $w_k(y)$ for any feasible SoC y. Let $w^{\text{Charge}}_k(y)$ be defined as the latest time at which charging can start from an SoC y at node i_k:

$$w^{\text{Charge}}_k(y) = \max_{y_1 \in [y, B]} \{w_k(y_1) - g_{i_k}(y, y_1)\}, \quad \forall y \in [0, B], \quad \forall k \in \{1, \ldots, n\} \tag{13}$$

or equivalently,

$$w^{\text{Charge}}_k(y) = \begin{cases} \max_{y_1 \in [y, B]} \{w_k(y_1) - h^{-1}_{i_k}(y_1) + h^{-1}_{i_k}(y)\}, & \text{if } i_k \in S, \\ w_k(y), & \text{otherwise,} \end{cases} \quad \forall y \in [0, B], \quad \forall k \in \{1, \ldots, n\}. \tag{14}$$

Note that by substituting w^{Charge}_k into (11), the optimization problem of deciding the optimal y_1 for a given y is removed as shown in (15):

$$w_k(y) = \mathrm{TW}^B_{i_{k+1}}\left(w^{\text{Charge}}_{k+1}(y - e_{i_k, i_{k+1}})\right) - t_{i_k, i_{k+1}}, \quad \forall y \in [0, B], \quad \forall k \in \{0, \ldots, n-1\}. \tag{15}$$

Hence, if w^{Charge}_{k+1} is known, calculating $w_k(y)$ is trivial. Further, if i_k is a customer node, calculating $w_k(y)$ is also trivial. We, therefore, focus on how to calculate w^{Charge}_k for $i_k \in S$ in Theorem 2. Similar to the forward case, we identify two possible strategies for determining the latest time at which recharging can start at node i_k with an SoC y. Either the vehicle charges the battery from an SoC y to y_1, making $w^{\text{Charge}}_k(y) = w_k(y_1) - h^{-1}_{i_k}(y_1) + h^{-1}_{i_k}(y)$, or it does not charge, making $w^{\text{Charge}}_k(y) = w_k(y)$. We again refer to these two policies as *Charge* and *No charge*, respectively.

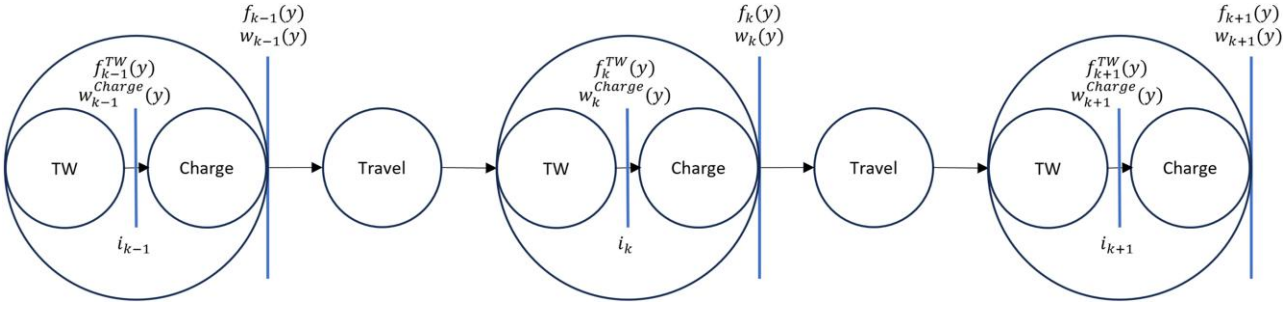
In this theorem, we denote by $\frac{df}{dy^-}$ the left-hand-side derivative of a function f and for ease of exposition, let $w_k(y) = -\infty$ and $\frac{dw_k(y)}{dy^-} = -\infty$ if $y \notin \mathit{domain}(w_k)$.

Theorem 2. *Let $i_k \in S$ be a recharging node and $y_1 \in (0, B]$ be an SoC value. If $w^{\text{Charge}}_k(y_1) = w_k(y_1)$ and $\frac{dw_k(y_1)}{dy^-} \le \frac{dh^{-1}_{i_k}(y_1)}{dy^-}$, then $w^{\text{Charge}}_k(y) = w_k(y)$ for all values $y \in [y_0, y_1]$, where $y_0 = \max\left\{0, \sup\left\{y' \mid y' < y_1, \frac{dw_k(y')}{dy^-} > \frac{dh^{-1}_{i_k}(y')}{dy^-}\right\}\right\}$. Otherwise, $w^{\text{Charge}}_k(y) = w^{\text{Charge}}_k(y_1) - h^{-1}_{i_k}(y_1) + h^{-1}_{i_k}(y)$ for all $y \in [y_0, y_1]$, where $y_0 = \max\left\{0, \max\left\{y' \mid y' < y_1, w_k(y') = w^{\text{Charge}}_k(y_1) - h^{-1}_{i_k}(y_1) + h^{-1}_{i_k}(y'), \frac{dw_k(y')}{dy^-} \le \frac{dh^{-1}_{i_k}(y')}{dy^-}\right\}\right\}$.*

The proof of Theorem 2 is provided in Online Appendix C. Note that that this proof does not rely on the concavity of function $h_i(t)$. Therefore, Theorem 2 may be applicable to more general charging functions.

Theorem 2 is used to compute w_k in Algorithm 2. The time complexity of Algorithm 2 is the same as that of Algorithm 1. Note that similar to Algorithm 1, we may simplify the calculation of y_0 on line 14 in Algorithm 2 to be $y_0 = \max\{0, \max\{y' \mid y' < y_1, w_k(y') = w^{\text{Charge}}_k(y_1) - h^{-1}_{i_k}(y_1) + h^{-1}_{i_k}(y')\}\}$ because it is a relaxation of the expression in Theorem 2. However, this relaxation may

Figure 6. (Color online) Illustration of a Sequence of Actions Taken Within and Between Nodes Along a Path



Note. Functions $f_k(y)$, $f^{\text{TW}}_k(y)$, $w_k(y)$, and w^{Charge}_k are also included to show where they are calculated.

cause $y_0 = y_1$ on line 10 in Algorithm 2 in the next iteration, leading to additional iterations in the while loop. In Online Appendix D, we provide an algorithm with linear time complexity tailored for the specific variant where all h_i functions are piecewise linear.

Algorithm 2 (Compute w_k^{Charge})

1: **function** CHARGE(w_k, h_{i_k})
2: $y_1 = B$
3: State ← No charge
4: $w_k^{\text{Charge}}(y_1) = w_k(y_1)$
5: **if** $\frac{dw_k(y_1)}{dy^-} > \frac{dh_{i_k}^{-1}(y_1)}{dy^-}$ **then**
6: State ← Charge
7: **end if**
8: **while** $y_1 > 0$ **do**
9: **if** State = No charge **then**
10: $y_0 = \max\left\{0, \sup\left\{y' \mid y' < y_1, \frac{w_k(y')}{dy^-} > \frac{dh_{i_k}^{-1}(y')}{dy^-}\right\}\right\}$
11: $\forall y \in [y_0, y_1] : w_k^{\text{Charge}}(y) = w_k(y)$
12: State ← Charge
13: **else**
14: $y_0 = \max\left\{0, \max\left\{y' \mid y' < y_1, w_k(y') = w_k^{\text{Charge}}(y_1) - h_{i_k}^{-1}(y_1) + h_{i_k}^{-1}(y'), \frac{dw_k(y')}{dy^-} \le \frac{dh_{i_k}^{-1}(y')}{dy^-}\right\}\right\}$
15: $\forall y \in [y_0, y_1] : w_k^{\text{Charge}}(y) = w_k^{\text{Charge}}(y_1) - h_{i_k}^{-1}(y_1) + h_{i_k}^{-1}(y)$
16: State ← No charge
17: **end if**
18: $y_1 \leftarrow y_0$
19: **end while**
20: **end function**

4. BP&C for E-VRPs with Heterogeneous Technologies and Nonlinear Recharging Functions

BP&C is arguably the most efficient method for solving many extensions of the VRP. BP&C combines branch and bound, column generation, and cutting planes in order to solve large-scale combinatorial optimization problems. In each node of the branch-and-bound tree, the linear relaxation, referred to as the master problem, is solved by column generation. This iterative method can solve linear programs containing a very large number of variables. At each iteration, it starts by solving a restricted master problem (RMP), where only a subset of the variables is included. Solving the RMP gives optimal dual-variable values that are used in a pricing problem, which is solved next. The pricing problem aims at finding columns (variables) that might improve the current RMP solution (i.e., columns with a negative reduced cost). If such columns are found, they are added to the RMP before starting a new iteration. Otherwise, the column generation algorithm stops as the current RMP solution is also optimal for the master problem. For recent advances in BP&C for VRPs, see Costa, Contardo, and Desaulniers (2019).

In our case, the RMP is the linear relaxation of (2)–(4), where only a subset of the feasible paths in Ω is present. The pricing problem is an elementary shortest-path problem with resource constraints (ESPPRC), where elementarity is imposed on the customer nodes only. Its objective is to find a feasible path $p \in \Omega$ that minimizes the reduced cost $\bar{c}_p = c_p - \sum_{i \in C} a_{ip}\alpha_i$, where α_i is the optimal dual value of Constraints (3) for customer $i \in C$. For ease of notation, let $\alpha_i = 0, \quad \forall i \in N \backslash C$.

The ESPPRC is solved by dynamic programming with a labeling algorithm. A labeling algorithm explores the graph G, searching for a least-cost path from the source node to the sink node. It does so by starting with a partial path only containing the source node and then gradually extending partial paths along all arcs. Each partial path is represented by a label. A label is a tuple containing essentially three elements: the index of the last node on the path, a pointer to the previous label, and a resource vector. The first two elements are only used to reconstruct the full path and are omitted from the following description, which focuses on the resource vector.

When a path is extended by appending an arc, the resource vector is updated according to resource extension functions. It is used for checking feasibility, keeping track of the cost of the partial path, and identifying dominance between labels. Dominance ensures that suboptimal partial paths are pruned from the search, avoiding enumeration of all feasible paths. If every feasible extension going from the current node of a label to the sink node is also feasible for another label and the extended path obtained from the former label is not cheaper than that yielded by the latter label, the former is dominated and may be pruned from the search. The reader is referred to Desrochers and Soumis (1988) and Irnich and Desaulniers (2005) for more in-depth information about labeling algorithms.

The remainder of this section is organized as follows. Section 4.1 presents the algorithm employed to tackle the pricing problem. Then, Sections 4.2 and 4.3 provide an overview of the cutting planes applied and the branching procedure used, respectively. Finally, Section 4.4 details the necessary adjustments to the solution method when aiming to minimize the total travel and charging time as in the E-VRP-NL of Montoya et al. (2017).

4.1. Solving the Pricing Problem

In this subsection, we present a bidirectional labeling algorithm (Righini and Salani 2006) for solving the pricing problem. Such an algorithm uses two monodirectional labeling algorithms going in opposite directions. The forward labeling algorithm is initiated at the source node, and the backward labeling algorithm is initiated

at the sink node. One of the resources (time in our implementation) is used to define a midway point ($\frac{\underline{T}_{d(s)}+\overline{T}_{d(e)}}{2}$ in our case), at which the monodirectional algorithms are stopped. Forward and backward labels are then merged into complete routes that are checked with regard to feasibility and reduced cost, and they are potentially added to the RMP. This method mitigates the effect of the exponential growth in the number of labels observed in complete monodirectional labeling algorithms.

The forward and backward labeling algorithms are presented in Sections 4.1.1 and 4.1.2, respectively, whereas the merging of partial paths is explained in Section 4.1.3. Acceleration techniques for the pricing problem are described in Section 4.1.4.

4.1.1. Forward Labeling. In the forward labeling algorithm, a label representing a path from $d(s)$ to a node i is a tuple $L_F = (i, d, \overline{c}, f, \mathcal{U})$. In this tuple, i denotes the last node of the path, and d denotes the vehicle load at this node. Its resource window is $[0, Q]$. The accumulated (reduced) cost is given by $\overline{c}$ and is unlimited. The resource f is special as it is a piecewise function encapsulating the interdependency between departure time from the current node and the SoC at departure. The former has a resource window $[\underline{T}_i, \overline{T}_i]$, and the latter has a resource window $[0, B]$. As such, $f = f_k(y)$ for the fixed path implied by the label (see Section 3). The last resource $\mathcal{U}$ is the set of customer nodes that are no longer reachable by a feasible extension of the partial path. Note that all resources need to be within their respective resource window at any point on the path for it to be feasible. Labels deemed infeasible are discarded without being extended. Hereafter, we denote by r_L the resource r of a label L.

The resource extension functions used to extend a label L_F along an arc (i, j) to yield a new label L'_F are as follows:

$$d_{L'_F} = d_{L_F} + q_j, \tag{16}$$

$$\overline{c}_{L'_F} = \overline{c}_{L_F} + c_{i,j} - \alpha_j, \tag{17}$$

$$f_{L'_F}(y) = \min_{y_0 \in [0,y]} \{\mathrm{TW}^{\mathrm{F}}_j(f_{L_F}(y_0 + e_{i,j})) + t_{i,j}) + g_j(y_0, y)\}, \quad \forall y \in [0, B], \tag{18}$$

$$\mathcal{U}_{L'_F} = \begin{cases} \mathcal{U}_{L_F} \cup U_F(L'_F) \cup \{j\}, & \text{if } j \in C, \\ \mathcal{U}_{L_F} \cup U_F(L'_F), & \text{otherwise.} \end{cases} \tag{19}$$

Extension functions (16) and (17) correspond to adjusting the vehicle load and the accumulated reduced cost, respectively. The extension function (18) is calculated as described in Section 3.1. Extension function (19) updates the set of unreachable customers, where $U_F(L'_F)$ gives the set of customers who are unreachable because of the other resources of L'_F. It is defined as $U_F(L'_F) = \{j' \in C \mid d_{L'_F} + q_{j'} > Q \vee f_{L'_F}(0) + t_{i,j'} > \overline{T}_{j'}\}$, where $i = i_{L'_F}$. Note that $f_{L'_F}(e_{i,j'}) + t_{i,j'}$ is not necessarily a valid lower bound on the arrival time at node j' because there might exist a subpath between nodes i and j' with an earlier arrival time (e.g., leaving node i earlier than $f_{L'_F}(e_{i,j'})$ with an SoC less than $e_{i,j'}$, visiting a fast charger, before continuing to node j'). Therefore, we instead use $f_{L'_F}(0) + t_{i,j'}$ as it is a guaranteed lower bound on the arrival time at node j' given that the triangle inequality holds with respect to time.

Extending label L_F along arc (i, j) gives a feasible label if criteria (20) and (21) hold:

$$j \notin \mathcal{U}_{L_F}, \tag{20}$$

$$f_{L_F}(e_{i,j}) + t_{i,j} \le \overline{T}_j. \tag{21}$$

In Condition (21), it is valid to use $f_{L_F}(e_{i,j})$ because this condition assesses the feasibility of a direct extension to node j.

Let L'_F and L''_F denote two labels representing paths ending at a node i. Then, label L'_F dominates label L''_F if

$$d_{L'_F} \le d_{L''_F}, \tag{22}$$

$$\overline{c}_{L'_F} \le \overline{c}_{L''_F}, \tag{23}$$

$$\mathcal{U}_{L'_F} \subseteq \mathcal{U}_{L''_F}, \tag{24}$$

$$f_{L'_F}(y) \le f_{L''_F}(y), \quad \forall y \in \mathit{domain}(f_{L''_F}). \tag{25}$$

Conditions (22) and (23) check that the accumulated demand and reduced cost of L'_F are less than or equal to those of L''_F, whereas Condition (24) ensures that all nodes reachable by L''_F are also reachable by L'_F. The resource extension functions for d, $\overline{c}$, and $\mathcal{U}$ are nondecreasing functions; hence, pair-wise comparisons of the labels are sufficient to check dominance with regard to these resources (Desaulniers et al. 1998). Given that the resource extension functions for resource f are not nondecreasing, a more elaborate check is needed. The Pareto frontiers representing the recharging potential for the labels are compared in Condition (25). In the special case of piecewise linear recharging functions, both $f_{L'_F}$ and $f_{L''_F}$ are piecewise linear functions with a limited domain as well. Hence, it is sufficient to check condition $f_{L'_F}(y) \le f_{L''_F}(y)$ for all values $y \in \mathit{domain}(f_{L''_F})$, where either function has a break or end point in order to check Condition (25).

4.1.2. Backward Labeling. The resources for the backward labeling are similar to the ones for the forward labeling. A label residing at node i writes as $L_B = (i, d, \overline{c}, w, \mathcal{U})$. The load on the partial path from node i to $d(e)$, excluding the demand from node i, is stored in d and has a resource window $[0, Q - q_i]$. Likewise, the path reduced cost is denoted by $\overline{c}$. Analogous to f in the forward labeling algorithm, w is a piecewise function that stores the interdependency between the latest feasible departure time from the current node and the SoC at departure. The former has a time window $[\underline{T}_i, \overline{T}_i]$, and the latter has a resource window $[0, B]$. The dependency

function is equal to $w_k(y)$, described in Section 3.2, for the fixed partial path starting at node i and ending at node $d(e)$. Resource $\mathcal{U}$ stores the set of backward unreachable customers.

When extending a label L_B backward along an arc (i, j) to yield a new label L'_B at node i, the following resource extension functions are applied:

$$d_{L'_B} = d_{L_B} + q_j, \tag{26}$$

$$\overline{c}_{L'_B} = \overline{c}_{L_B} + c_{i,j} - \alpha_j, \tag{27}$$

$$w_{L'_B} = \mathrm{TW}_j^B\left(\max_{y_1 \in [y - e_{i,j}, B]} \{w_{L_B}(y_1) - g_j(y - e_{i,j}, y_1)\}\right) - t_{i,j}, \quad \forall y \in [0, B], \tag{28}$$

$$\mathcal{U}_{L'_B} = \begin{cases} \mathcal{U}_{L_B} \cup U_B(L'_B) \cup \{j\}, & \text{if } j \in C, \\ \mathcal{U}_{L_B} \cup U_B(L'_B), & \text{otherwise.} \end{cases} \tag{29}$$

Extension functions (26) and (27) adjust the vehicle load and the accumulated reduced cost. The extension function (28) is calculated as described in Section 3.2. Extension function (29) updates the set of unreachable customer nodes. Specifically, the set of customers who cannot be reached because of the resource values of label L'_B is given as $U_B(L'_B) = \{j' \in C \mid d_{L'_B} + q_{j'} + q_i > Q \vee w_{L'_B}(B) - t_{j',i} < \underline{T}_{j'}\}$, where $i = i_{L'_B}$. For an extension of label L_B along arc (i, j) to be feasible, Conditions (30) and (31) must hold:

$$i \notin \mathcal{U}_{L_B}, \tag{30}$$

$$w_{L_B}(B - e_{i,j}) - t_{i,j} \geq \underline{T}_i. \tag{31}$$

Let L'_B and L''_B denote two labels representing paths ending at a node i. Then, label L'_B dominates label L''_B if

$$d_{L'_B} \leq d_{L''_B}, \tag{32}$$

$$\overline{c}_{L'_B} \leq \overline{c}_{L''_B}, \tag{33}$$

$$\mathcal{U}_{L'_B} \subseteq \mathcal{U}_{L''_B}, \tag{34}$$

$$w_{L'_B}(y) \geq w_{L''_B}(y), \quad \forall y \in \mathit{domain}(w_{L''_B}). \tag{35}$$

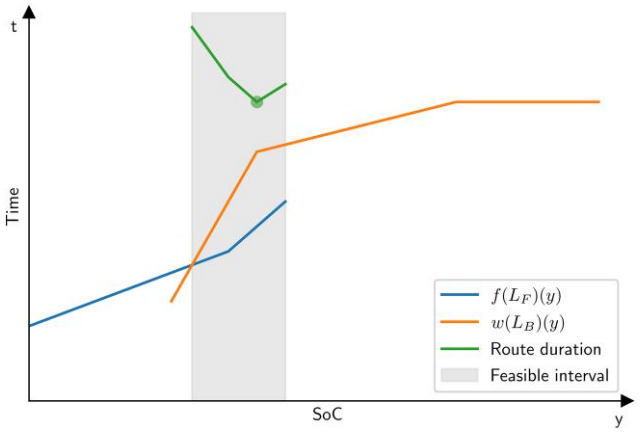
As in the forward labeling algorithm, Conditions (32)–(34) are pair-wise comparisons of the resources as the corresponding resources and their resource extension functions are, or can be transformed into, nondecreasing functions. Condition (35) is the parallel to Condition (25) applied in the forward labeling algorithm.

4.1.3. Merging Partial Paths. The requirements for merging of a forward label, L_F, and a backward label, L_B, both ending at a node i to be feasible are given by conditions $d_{L_F} \geq d_{L_B} \leq Q$, $\mathcal{E}_{L_F} \cap \mathcal{E}_{L_B} \subseteq \{i\}$, and $\exists y \in \mathit{domain}(f_{L_F}) \cap \mathit{domain}(w_{L_B})$ such that $(f_{L_F}(y) \leq w_{L_B}(y))$. The first condition checks vehicle capacity feasibility. Customer elementarity is checked by the second condition, where $\mathcal{E}_L$ denotes the set of customer nodes visited by the path represented by a label L. These sets are computed for every partial path found by the forward or backward labeling algorithms prior to merging partial paths. The existence of a feasible schedule in terms of time windows and battery energy is enforced by the third condition. The check is illustrated in Figure 7, where the condition states that the area marked as the feasible interval must be nonempty. The reduced cost is checked by $\overline{c}_{L_F} + \overline{c}_{L_B} < 0$.

4.1.4. Acceleration Techniques for the Pricing Problem. In addition to using bidirectional labeling, several well-known techniques are used to improve the overall efficiency of the solution method. For more details on these techniques, see the survey by Costa, Contardo, and Desaulniers (2019).

A combination of *ng*-path relaxation (Baldacci, Mingozzi, and Roberti 2011) and decremental state-space relaxation (Righini and Salani 2008) is used in order to improve the efficiency of the labeling algorithm. In the *ng*-path relaxation, a predefined neighborhood $\mathcal{N}_i$ is associated with each node i. Then, a cycle $(i_k, \ldots, i_{k+m})$, with $i_k = i_{k+m} \in C$, is allowed if there exists at least one node i_{k+j}, $j \in \{1, \ldots, m-1\}$, for which $i_k \notin \mathcal{N}_{k+j}$. Decremental state-space relaxation can be seen as lazy constraints. Nonelementary paths are initially allowed, but if a path that is nonelementary with respect to node i is optimal, adjustments to the labeling algorithm are made such that all paths visit node i at most once in later runs of the labeling algorithm. In the labeling algorithm employed in this work, we initially include the $\max\{8, \lfloor 0.2 \mid N \mid \rfloor\}$-nearest customer nodes to node i into set $\mathcal{N}_i$. In every iteration of the column generation

Figure 7. (Color online) Example of a Combination of f_{L_F} and w_{L_B} Functions



Notes. When minimizing the total travel and charging time for the E-VRP-NL, the SoC value within the feasible interval that minimizes the route duration must be calculated as described in optimization Problem (42). *Route duration* is found by calculating the objective function given in (42) for all feasible y values, where f_{L_F} or w_{L_B} has a break or end point. The optimal solution and objective value are both indicated by a dot.

algorithm, we check if the linear relaxation solution contains a variable λ_p with a positive value such that the associated path p visits a customer node i more than once. Any such node found is then appended to the neighborhood, $\mathcal{N}_j$, of every other node $j \in N$. This operation is performed a maximum of 10 times. Any subsequent elementarity violation is handled through the branching process.

The pricing problem is usually solved heuristically and only solved exactly if the heuristics do not find any negative reduced cost columns. The heuristics implemented solve the pricing problem on a reduced network, where only some arcs are kept. We have adopted the method used for choosing which arcs to keep from Desaulniers, Lessard, and Hadjar (2008). Outgoing arcs from node i are sorted according to their reduced cost, $c_{ij} - \alpha_j$, and given a rank. This is done equivalently for incoming arcs to node i, and hence, each arc (i, j) is associated with two ranks: one where it is considered outgoing from node i and one where it is considered incoming to node j. Only arcs where either rank is less than some predefined limit are kept in the network. Two such heuristics are used: one where the limit is three and one where the limit is six. The heuristic with a limit of three always employs a monodirectional labeling algorithm, whereas the heuristic with a limit of six utilizes the bidirectional labeling algorithm.

4.2. Cutting Planes

Subset row cuts (Jepsen et al. 2008) with subsets $\mathcal{W} \subset C$ of size 3 are used to strengthen the dual bound. They can be stated as $\sum_{p\in\Omega} \left\lfloor \frac{\sum_{i\in\mathcal{W}} a_{ip}}{2} \right\rfloor \lambda_p \leq 1$ and are separated by complete enumeration. In order to mitigate the negative effect in the pricing problem when using subset row cuts, limited memory subset row cuts are used (Pecin et al. 2017). In the pricing problem, the cuts are handled by introducing a binary resource for each cut, which affects the labeling extension, dominance checking, and label merging.

4.3. Branching

The branching strategy comprises three levels. The entity with the highest priority is chosen among the entities with a fractional value. Foremost is the consideration of the number of routes utilized (i.e., $\sum_{p\in\Omega}\lambda_p$). Following this, the secondary priority involves the count of recharging operations for each recharging node expressed as $\sum_{p\in\Omega} a_{ip}\lambda_p$ for recharging node i. Lastly, branching is performed on the edge in G with the most fractional flow. Branching is enforced by adding constraints to the RMP. These additional constraints introduce dual variables that are considered in the pricing problem. The exploration of the branch-and-bound tree follows a best-first approach.

4.4. Minimizing Total Time

The E-VRP-NL, as described by Montoya et al. (2017), is similar to the E-VRPTW-NL. The main difference is that the E-VRP-NL objective is minimizing the total time spent by all vehicles rather than minimizing the total traveling cost. Additionally, the E-VRP-NL considers uncapacitated vehicles and a maximum route duration instead of customer time windows. The proposed BP&C algorithm is able to cope with both features by setting the values for the vehicle capacity and the time windows sufficiently low/high.

The change in objective may seem rather minor. However, it affects the labeling algorithm because the reduced cost is now indirectly affected by the SoC. Hence, besides having a Pareto front describing the relationship between SoC and time, each label also represents a Pareto front describing the relationship between SoC and reduced cost. However, the SoC/reduced cost Pareto front does not need to be stored explicitly. The negatives of the accumulated dual values are stored in the resources c_{LF} and c_{LB} in forward and backward labeling, respectively. Then, the Pareto front describing the relationship between the SoC and the reduced cost can be obtained by $\overline{c}_{L_F}(y) = f_{L_F}(y) + c_{L_F}$ and $\overline{c}_{L_B}(y) = \overline{T}_{d(e)} - w_{L_B}(y) + c_{L_B}$ for forward and backward labels, respectively. Hence, the difference in the two Pareto fronts given by each label is a constant.

The resource extension functions for c_{L_F} and c_{L_B} for forward and backward labeling when extending along arc (i, j) are given by (36) and (37), respectively:

$$c_{L'_F} = c_{L_F} - \alpha_j, \tag{36}$$

$$c_{L'_B} = c_{L_B} - \alpha_j. \tag{37}$$

The changes in the dominance criteria are as follows. Requirements (23) and (33) are replaced by (38) and (39):

$$\overline{c}_{L'_F}(y) \leq \overline{c}_{L''_F}(y), \qquad \forall y \in domain(f_{L''_F}), \tag{38}$$

$$\overline{c}_{L'_B}(y) \leq \overline{c}_{L''_B}(y), \qquad \forall y \in domain(w_{L''_B}). \tag{39}$$

However, one can exploit the fact that the difference between the two Pareto fronts given by a label is a constant and check dominance with regard to the two resources simultaneously. The combined dominance checks are given in requirements (40) and (41):

$$f_{L'_F}(y) - f_{L''_F}(y) \leq \min\{0, c_{L''_F} - c_{L'_F}\}, \quad \forall y \in domain(f_{L''_F}), \tag{40}$$

$$w_{L'_B}(y) - w_{L''_B}(y) \geq \max\{0, c_{L'_B} - c_{L''_B}\}, \quad \forall y \in domain(w_{L''_B}). \tag{41}$$

Finding the minimum route duration and hence, the cost is not as trivial. It is given by the minimization problem:

$$\begin{aligned} c^* = \min \quad & \overline{T}_{d(e)} + f_{L_F}(y) - w_{L_B}(y), \\ \text{s.t.} \quad & y \in domain(f_{L_F}) \cap domain(w_{L_B}), \\ & f_{L_F}(y) \leq w_{L_B}(y). \end{aligned} \tag{42}$$

Let the node at which the partial paths are merged be denoted i. The variable y represents the SoC at

departure from node i. Thus, the objective is to minimize the time that it takes for the forward partial path to depart from node i with an SoC y plus the time it takes for the backward partial path to end at node $d(e)$ if departing node i with an SoC y.

Additionally, for calculating the route reduced cost, one needs to account for the accumulated dual values: $c^* + c_{L_F} + c_{L_B}$. An example of the minimum route duration as a function of the SoC at departure from node i is shown in Figure 7, which shows the entire solution space and corresponding objective values of the minimization Problem (42).

5. Computational Study

In this section, we report various numerical results obtained with the BP&C solution method. First, we present in Section 5.1 the new benchmark instances of the E-VRPTW-NL with heterogeneous technologies. Then, the effectiveness of the BP&C algorithm is investigated in Section 5.2 by comparing the results obtained on the E-VRPTW-L and the E-VRP-NL with results from the literature and measuring its performance on the new instances proposed for the E-VRPTW-NL with heterogeneous recharging technologies. Furthermore, the performance difference between using a bidirectional labeling algorithm and a monodirectional labeling algorithm for solving the pricing problem is investigated. Finally, the applicability of different simplifications to the recharging functions is discussed in Section 5.3, with the aim of providing insight into which simplifications are viable for different problem characteristics.

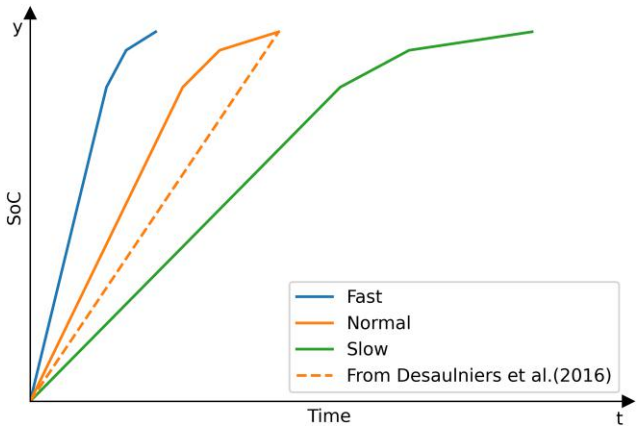
The BP&C algorithm, including the branch-and-bound tree search and the labeling algorithms, was implemented from scratch in C++ and compiled with GNU GCC 10.3.0. The RMP is solved using Gurobi 9.1. All computational tests were run on a Lenovo ThinkSystem SD530 running CentOS 7.9.2009 with a 3.5-GHz Intel Xeon Gold 6144 CPU and 384 GB of RAM.

5.1. New E-VRPTW-NL Benchmark Instances

The E-VRPTW-NL instances generated and used for our tests are based on the instances of Desaulniers et al. (2016) and Montoya et al. (2017). The instances of Desaulniers et al. (2016) are a slight modification of the instances proposed by Schneider, Stenger, and Goeke (2014), which in turn, are based on the well-known Solomon (1987) instances. The locations of the depot, customers, and recharging stations are taken from Desaulniers et al. (2016), whereas customer demands and service times retain their original values from Solomon (1987). The time windows are adopted from Schneider, Stenger, and Goeke (2014). These instances are classified by the number of customers (25, 50, or 100); the width of the time windows, which are either narrow (type 1) or wide (type 2); and the customer location distribution, which may be random, clustered, or a combination of both. The instances include 21 recharging stations, a number derived from Schneider, Stenger, and Goeke (2014), ensuring consistency with earlier works. Although an unlimited number of vehicles is assumed in the instances, the proposed algorithm can easily accommodate a limitation on the number of vehicles.

The recharging functions are adapted versions of those used in the instances of Montoya et al. (2017), who propose three nonlinear recharging functions: *fast*, *normal*, and *slow*. These functions are designed to emulate chargers with capacities of 44, 22, and 11 kW, respectively. For each recharging node in the data set, the recharging technology is set to be fast, normal, or slow with equal probability. The functions are scaled in the battery dimension to match the battery capacity of the vehicles, which is given by the battery capacity from the instances of Desaulniers et al. (2016). The normal recharging function is scaled along the time axis so that fully charging an empty battery takes the same time as in the original instances proposed by Desaulniers et al. (2016). The fast and slow recharging functions are then scaled along the time axis so that the time ratios between the chargers are the same as in Montoya et al. (2017). The average number of recharging operations per route is approximately 1.10 in the results reported by Desaulniers et al. (2016) and 1.02 in the results reported by Montoya et al. (2017). Given the similarity in instance characteristics, we expect comparable values for the average number of recharging operations in our experiments. The relationship between the charging function from Desaulniers et al. (2016) and the resulting recharging functions is illustrated in Figure 8. We follow the convention from earlier works on E-VRPTWs that forbids traveling directly between two recharging nodes. However, it is not a requirement of our solution method.

Figure 8. (Color online) Illustration of the Original Charging Function from Desaulniers et al. (2016) and the Adapted Nonlinear Recharging Functions



5.2. Algorithmic Performance

This section presents the numerical results obtained by testing the presented BP&C method. First, we compare the performance of this method against existing solution methods for the E-VRPTW-L and the E-VRP-NL in Sections 5.2.1 and 5.2.2, respectively. The goal of the comparison is to investigate how the proposed method performs, relative to known methods, on less complex variants of the problem. Then, we report in Section 5.2.3 numerical results obtained by the proposed method on the new E-VRPTW-NL instances. In this section, average results are reported. Detailed results can be found in Online Appendix F.

5.2.1. Comparison with Desaulniers et al. (2016) for the E-VRPTW-L. The E-VRPTW-NL with heterogeneous recharging technologies is a more general version of the E-VRPTW-L with a single recharging technology, and hence, the presented solution method is also applicable for this problem. Here, we compare the performance of our solution method with the BP&C method developed by Desaulniers et al. (2016) for the variant denoted EVRPTW-MP in their paper. Note that the algorithm of Desaulniers et al. (2016) was later improved by adding variable fixing (Desaulniers, Gschwind, and Irnich 2020), a methodology that could also be added to our algorithm. Results for this problem are also presented by Lera-Romero, Miranda Bront, and Soulignac (2024). However, their results are obtained on a network with preprocessed time windows for the VRPTW, which are not valid for the E-VRPTW. Indeed, their time window reduction does not consider the eventual need to charge along an arc, possibly excluding feasible paths. A detailed explanation, including an example, is provided in Online Appendix E.

Table 1 presents aggregated numerical results for both our solution method and that of Desaulniers et al. (2016) for the E-VRPTW-L, with a maximum computing time of one hour (3,600 seconds). Columns #Cust., Type, and #Inst. indicate the number of customers, the instance type, and the number of instances, respectively, for a given aggregation of instances. The numbers of instances solved to optimality by each method are shown in the columns labeled #Solved, whereas the average computing times (in seconds) are given in the columns labeled Time. For instances not solved to optimality, the maximum time is used in the calculation.

Table 1. Results on the E-VRPTW-L Instances

			Desaulniers et al. (2016)		Our method	
#Cust.	Type	#Inst.	#Solved	Time (seconds)	#Solved	Time (seconds)
25	1	29	29	7.1	29	7.6
25	2	27	26	179.6	27	20.9
50	1	29	26	713.6	26	524.1
50	2	27	19	1,524.9	23	1,048.9
100	1	29	10	2,503.3	11	2,496.7
100	2	27	8	2,782.2	4	3,204.7

Notes. Columns #Cust., Type, and #Inst. indicate the number of customers, the instance type, and the number of instances, respectively, for a given aggregation of instances. The number of instances solved to optimality by each method is shown in the columns labeled #Solved, whereas the average computing time (in seconds) is given in the columns labeled Time.

The results show that our BP&C algorithm solves one more of the 25-customer instances and four more of the 50-customer instances than the algorithm of Desaulniers et al. (2016) does, but the latter solves three more of the 100-customer instances within one hour. Notice that the CPU utilized by Desaulniers et al. (2016) has a lower single-thread rating (https://www.cpubenchmark.net) than the one used in this paper (2,173 versus 2,509). We conclude that despite the fact that our method is designed to solve a more general problem, our results are competitive with those of Desaulniers et al. (2016) for the E-VRPTW-L.

5.2.2. Comparison with Froger et al. (2019) and Kancharla and Ramadurai (2020) for the E-VRP-NL. The proposed method, with the adjustments described in Section 4.4, has been tested on the instances from Montoya et al. (2017) and compared with the current state-of-the-art solution methods devised by Froger et al. (2019) and Kancharla and Ramadurai (2020). In these instances, traveling between two recharging nodes is allowed. Like in Montoya et al. (2017), Froger et al. (2019), and Kancharla and Ramadurai (2020) for the exact algorithms, the maximum computing time was set to three hours.

The numerical results are presented in Table 2, where the column headings all have the same interpretation as described above. These results clearly indicate that our BP&C method significantly improves upon the current state-of-the-art methods as all but one of the 20-customer instances are solved to optimality, whereas the methods by Froger et al. (2019) and Kancharla and Ramadurai (2020) solve only five of them. The presented method is also able to solve 10 of the 40-customer instances, leading to a total of 24 new benchmark instances solved to optimality. The computing times are also significantly lower, reducing the computing times by factors of 44 and 6 for the 10- and 20-customer instances, respectively. It should be noted that the single-threaded CPU ratings of the CPUs used by Froger et al. (2019) and Kancharla and Ramadurai (2020) are 1,493 and 1,183, respectively, compared with 2,509 in our case. However, this can only explain a small part of the performance gap displayed by the results.

5.2.3. Numerical Results for the New E-VRPTW-NL Instances. The algorithmic performance is studied further by solving the E-VRPTW-NL instances described in Section 5.1. The maximum computing time was set to

Table 2. Results on the E-VRP-NL Instances of Montoya et al. (2017)

		Froger et al. (2019)		Kancharla and Ramadurai (2020)		Our method	
#Cust.	#Inst.	#Solved	Time (seconds)	#Solved	Time (seconds)	#Solved	Time (seconds)
10	20	20	229.5	20	80.1	20	1.8
20	20	5	8,222.2	5	8,893.0	19	1,211.1
40	20	0	10,800.0	0	10,800.0	10	5,888.6

Notes. Columns #Cust., and #Inst. indicate the number of customers and the number of instances, respectively, for a given aggregation of instances. The number of instances solved to optimality by each method is shown in the columns labeled #Solved, whereas the average computing time (in seconds) is given in the columns labeled Time.

one hour. The aggregated computational results when using a bidirectional labeling algorithm for solving the pricing problem are presented in Table 3, where the columns #Cust., Type, #Inst., #Solved, and Time have the same interpretation as above. Columns #Nodes and #SR cuts indicate the number of branch-and-bound nodes explored and the number of subset row cuts applied, respectively. The columns under #Routes with x recharge oper. show the average numbers of routes in the optimal solution with x recharging operations, where x is indicated by the number below. Only instances solved to optimality are included in this calculation.

The numerical results show that the algorithm is able to solve instances with up to 100 customers. As expected, the results for instances with narrow time windows (type 1) are better than those for the instances with wide time windows (type 2). Furthermore, a comparison with Table 1 indicates that the computing time used by the proposed method only increases slightly when the instances contain heterogeneous nonlinear recharging functions compared with a single linear recharging function.

Further analysis of the solutions shows that even though the average route length increases for the instances with wide time windows, the average number of recharging operations per route actually decreases. This is likely because the vehicle is able to recharge more at each visit to a recharging station because it does not have to abort the charging to reach a customer before the end of its time window. For the instances with narrow time windows, we see that it is optimal to do many short partial recharging operations as opposed to fewer long ones.

Table 4 reports the computational results when the pricing problem is solved by a monodirectional forward labeling algorithm. The columns are the same as in Table 3. Comparing the results with those presented in Table 3, we see that using a bidirectional labeling algorithm for solving the pricing problem exactly outperforms using a monodirectional labeling algorithm. Twelve instances are only solved to optimality when using bidirectional labeling, whereas no instance is only solved when using monodirectional labeling. Further, the computing times are, on average, significantly lower.

5.3. Recharging Function Simplifications

Montoya et al. (2017) showed that modeling the recharging process with a piecewise linear function gives an accurate estimate of the time that it takes to recharge. As reviewed in Section 1.1, there exist several effective solution methods that assume either homogeneous or heterogeneous linear charging functions or a single nonlinear charging function. It is thus interesting to analyze how different simplifications of the recharging functions affect the feasibility and the cost of the vehicle

Table 3. Results on the E-VRPTW-NL Instances When Using Bidirectional Labeling

							#Routes with x recharge oper.			
#Cust.	Type	#Inst.	#Solved	Time (seconds)	#Nodes	#SR cuts	0	1	2	≥3
25	1	29	29	9.8	1.4	7.9	0.76	2.48	1.48	0.21
25	2	27	27	40.0	1.0	3.4	1.81	0.81	0.30	0.00
50	1	29	27	499.0	22.9	24.6	1.22	3.74	2.48	0.22
50	2	27	22	1,152.6	1.3	8.4	2.36	1.68	0.27	0.05
100	1	29	8	2,800.9	66.8	71.1	3.88	5.38	3.75	0.88
100	2	27	4	3,205.2	2.6	9.2	3.50	2.25	0.50	0.00

Notes. Columns #Cust., Type, and #Inst. indicate the number of customers, the instance type, and the number of instances, respectively, for a given aggregation of instances. The number of instances solved to optimality by each method is shown in the column labeled #Solved, whereas the average computing time (in seconds) is given in the column labeled Time. Columns #Nodes and #SR cuts indicate the number of branch-and-bound nodes explored and the number of subset row cuts applied, respectively. The columns under #Routes with x recharge oper. show the average number of routes in the optimal solution with x recharging operations, where x is indicated by the number below.

Table 4. Results on the E-VRPTW-NL Instances When Using Monodirectional Labeling

#Cust.	Type	#Inst.	#Solved	Time (seconds)	#Nodes	#SR cuts
25	1	29	29	37.0	1.6	8.4
25	2	27	27	66.3	1.0	2.4
50	1	29	23	1,152.1	17.2	23.3
50	2	27	18	1,728.3	0.9	6.6
100	1	29	6	2,981.0	4.6	37.3
100	2	27	2	3,489.9	1.3	2.4

Notes. Columns #Cust., Type, and #Inst. indicate the number of customers, the instance type, and the number of instances, respectively, for a given aggregation of instances. The number of instances solved to optimality by each method is shown in the column labeled #Solved, whereas the average computing time (in seconds) is given in the column labeled Time. Columns #Nodes and #SR cuts indicate the number of branch-and-bound nodes explored and the number of subset row cuts applied, respectively.

routes for the E-VRPTW-NL with heterogeneous nonlinear recharging functions.

Five simplifications of the recharging process have been tested and analyzed. The simplifications are denoted NL1, L1 secant, L3 secant, L1 extrapolation, and L3 extrapolation, where L denotes that the recharging functions are linear, whereas NL indicates that they are nonlinear (piecewise linear). The number represents the number of recharging technologies. If there is only one, then it is based on the normal-type recharging function. In the cases where the recharging functions are purely linear, they are found by taking the secant or the fastest line segment from the corresponding nonlinear function used and extrapolated up to the maximum battery capacity. Hence, the L1 secant produces the instances of Desaulniers et al. (2016), and L3 extrapolation is the only one yielding a relaxation of the original problem. The different recharging function versions are illustrated in Figure 9.

For each simplification, we have run our algorithm on all of the test instances presented in Section 5.1. Each vehicle route in each optimal solution has then been re-evaluated by fixing the node sequence of the route and checking if a feasible charging schedule exists with respect to the charging technologies of each node in the original instance. An example of such a re-evaluation is shown in Figure 10, where the development of the vehicle's SoC over time for an optimal route for the simplification L3 extrapolation is compared with the charging schedule obtained by using the original charging functions. In Figure 10, we see that the simplification overestimates how much the vehicle charges at the first charging station before having to leave to reach the next customer within its time window. This leads to the vehicle battery becoming empty before reaching the second charging node, leading to an infeasible solution.

Table 5 shows the number of routes (#Routes) and solutions (#Solutions) found when aggregating the results based on the number of customers, the type of time windows, and the simplification of the recharging functions. Only instances solved to optimality are included. The rows denoted %Routes feas. and %Sol. feas. provide the percentages of the routes and solutions that are feasible when evaluated with the original recharging functions, respectively. In Table 6, columns Avg. and Max. denote the average and maximum objective value increases in percentages, respectively, when comparing the optimal objective value of the simplified problem and the original problem. Similarly, the columns denoted Opt. indicate the number of times that the optimal solution to the given simplification is feasible and has the same objective value as the original instance. Only those instances where both the simplified problem and the original problem are solved to optimality and the simplified solution is feasible when evaluated with the original recharging functions are included. The number of such instances is shown in the #Inst. columns.

A few insights can be seen from Tables 5 and 6. First, one can see that all simplifications usually yield feasible

Figure 9. (Color online) Different Recharging Functions Used During Testing

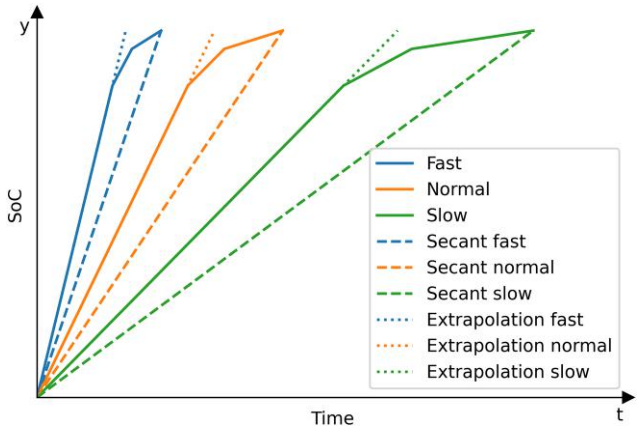


Figure 10. (Color online) The Development of the SoC over Time for a Route Given That Recharging Is Modeled Using the L3 Extrapolation and NL3 Approaches

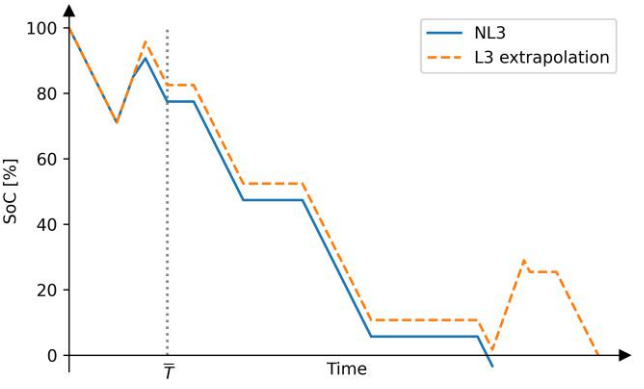


Table 5. Results for the Different Simplifications with Regard to Feasibility

	25 Customers		50 Customers		100 Customers	
Simplification	Type 1	Type 2	Type 1	Type 2	Type 1	Type 2
L1 secant						
#Routes	148	79	208	100	153	25
%Routes feas.	96.6	100.0	90.9	100.0	91.5	100.0
#Solutions	29	27	26	23	11	4
%Sol. feas.	82.8	100.0	50.0	100.0	18.2	100.0
L3 secant						
#Routes	147	79	197	92	139	31
%Routes feas.	99.3	100.0	99.0	100.0	98.6	100.0
#Solutions	29	27	25	21	10	5
%Sol. feas.	96.6	100.0	92.0	100.0	80.0	100.0
L1 extrapolation						
#Routes	145	79	203	96	152	35
%Routes feas.	93.1	98.7	86.2	100.0	88.8	94.3
#Solutions	29	27	26	22	11	6
%Sol. feas.	69.0	96.3	38.5	100.0	18.2	66.7
L3 extrapolation						
#Routes	142	79	208	96	111	31
%Routes feas.	97.2	98.7	96.6	100.0	100.0	100.0
#Solutions	29	27	27	22	8	5
%Sol. feas.	89.7	96.3	74.1	100.0	100.0	100.0
NL1						
#Routes	145	79	195	97	166	35
%Routes feas.	93.1	100.0	87.2	100.0	89.2	97.1
#Solutions	29	27	25	22	12	6
%Sol. feas.	69.0	100.0	40.0	100.0	16.7	83.3
NL3						
#Routes	143	79	207	96	111	25
%Routes feas.	100.0	100.0	100.0	100.0	100.0	100.0
#Solutions	29	27	27	22	8	4
%Sol. feas.	100.0	100.0	100.0	100.0	100.0	100.0

Notes. This table shows the number of routes (#Routes) and solutions (#Solutions) found when aggregating the results based on the number of customers, type of time windows, and the simplification of the recharging functions. Only instances solved to optimality are included. The rows denoted %Routes feas. and %Sol. feas. provide the percentages of the routes and solutions, resepectively, that are feasible when evaluated with the original recharging functions.

solutions when the time windows are wide. Additionally, Table 6 shows that the objective values for the type 2 instances only increase for the L1 secant and L3 secant simplifications, where the maximum increases are 0.02% and 1.42%, respectively. Hence, if the instances that one faces have wide time windows, there is little benefit of solving the problem with heterogeneous nonlinear recharging functions as a simplification might give an equally good outcome at a lower computational cost and less implementation effort.

Second, with regard to feasibility, it seems that keeping the recharging technologies heterogeneous is more important than how each of them is represented, and the secant simplifications give, in general, a feasible

Table 6. Increase in Objective Value for the Different Simplifications

	Type 1				Type 2			
Simplification	Avg. (%)	Max. (%)	Opt. (%)	#Inst.	Avg. (%)	Max. (%)	Opt. (%)	#Inst.
L1 secant	0.67	3.39	26.32	38	0.00	0.02	98.11	53
L3 secant	0.52	3.13	32.73	55	0.03	1.42	92.31	52
L1 extrapolation	0.14	1.27	65.63	32	0.00	0.00	100.00	52
L3 extrapolation	0.00	0.00	100.00	53	0.00	0.00	100.00	52
NL1	0.18	1.27	62.50	32	0.00	0.00	100.00	52

Notes. Columns Avg. and Max. denote the average and maximum objective value increases in percentage, respectively, when comparing the optimal objective value of the simplified problem and the original problem. Similarly, the columns denoted Opt. indicate the number of times that the optimal solution to the given simplification is feasible and has the same objective value as the original instance. Only those instances where both the simplified problem and the original problem are solved to optimality and the simplified solution is feasible when evaluated with the original recharging functions are included. The number of such instances is shown in the #Inst. columns.

solution more often than the simplifications using extrapolation from the fastest slope. This might be because of the robustness that some of the simplifications offer with regard to the original recharging functions. Charging on intervals where the original recharging function has a higher recharging rate than the simplification introduces robustness through slack in the scheduling. The simplification overestimates the charging time, and the excess time might be spent later on the path whenever the simplification underestimates the recharging time. However, the excess time might also be lost because of having to wait for time windows to open.

It is well known that introducing robustness in solutions often come at the expense of a worse objective value. Table 6 supports this statement as the secant simplifications give more expensive solutions than the simplifications using extrapolation from the fastest slope. Notably, L3 extrapolation is a relaxation of the original problem, so whenever it is solved to optimality and finds a feasible solution to the original problem, the solution is going to be optimal for the original problem too. That simplification is also the only one that under no circumstances accumulates excess time and hence, robustness because of the recharging modeling.

To further investigate how the infeasible solutions from each simplification differ from the solutions produced with NL3, we present a more detailed comparison in Table 7. A VRP solution can be thought of as two simultaneous decisions: (1) how to partition the set of customers into subsets that are served on the same vehicle route and (2) in what sequence the customers on each route are visited. In this analysis, we consider only instances where both algorithms with the simplification and with NL3 have solved the instance to optimality and the solution found with the simplification is infeasible when evaluated with the real recharging functions. For each simplification, Table 7 gives the number of instances where both the simplification and the original problem are solved to optimality (#Inst.), the number of those instances where the algorithm with the simplification provides an infeasible solution to the original problem (#Infeasible solutions), and the percentage of these where the customer partition in the optimal solution obtained with the simplification is the same as in the NL3 optimal solution (Same customer partition (%)). Further, for those instances where the partition differs, we calculate the percentage of customers who must be moved to recreate the partition given by the optimal solution to NL3 (Min customer moves (%) in Table 7). This is done by solving a matching problem. Finally, the percentage of the instances where all of the customer sequences are the same as in the NL3 solution is given (Same customer sequences (%) in Table 7).

Table 7 shows that in almost 80% of the instances where the simplifications give infeasible solutions, (some of) the routes serve different customers. Even for L3 extrapolation, which is a relaxation of NL3, the customers are partitioned differently in almost half of the instances with an infeasible solution. Further, we see that, on average, more than 10% of the customers have moved to a different route, which means that for an instance with 100 customers, more than 10 customers are served by a different route. This clearly indicates that the structure of the solutions can change significantly when the recharging model is simplified. An additional 7.8% (= 23.1 − 15.3%) of the instances have routes with the same customers, but for at least one route, the customer sequence is different. For the remaining 15.3% of the instances, both the partitioning and the customer sequences are the same. For these instances, the positioning of the recharging stations, the selection of charging stations, or both differ.

To summarize, the numbers in Tables 5 and 6 indicate that the modeling of the recharging process is less important when the time windows are wide; hence, using a single linear recharging function might give the best results. When the time windows are narrow, one can use the L3 secant simplification if finding a good feasible solution is acceptable. If an optimal solution is

Table 7. Statistics on Infeasible Solutions from the Simplifications

Simplification	#Inst.	#Infeasible solutions	Same customer partition (%)	Min customer moves (%)	Same customer sequences (%)
L1 secant	114	23	4.3	16.2	0.0
L3 secant	112	5	20.0	5.5	20.0
L1 extrapolation	115	31	19.4	13.2	9.7
L3 extrapolation	116	11	54.5	16.0	36.4
NL1	113	29	17.2	13.2	10.3
Average			23.1	12.8	15.3

Notes. For each simplification, this table gives the number of instances where both the simplification and the original problem are solved to optimality (#Inst.), the number of those instances where the algorithm with the simplification provides an infeasible solution to the original problem (#Infeasible solutions), and the percentage of these where the customer partition in the optimal solution obtained with the simplification is the same as in the NL3 optimal solution (Same customer partition (%)). Further, for those instances where the partition differs, we calculate the percentage of customers who must be moved to recreate the partition given by the optimal solution to NL3 (Min customer moves (%)). This is done by solving a matching problem. Finally, the percentage of the instances where all of the customer sequences are the same as in the NL3 solution is given (Same customer sequences (%)).

sought, L3 extrapolation might be the best option. Further, the results from Table 7 show that solving the instances with simplified recharging functions often leads to different partitions and/or sequences of customers, indicating that solving a simplification of the E-VRPTW-NL in combination with an exact algorithm for scheduling the recharging is not a viable strategy to solve the full problem.

6. Concluding Remarks

In this paper, we present an exact solution method for the E-VRPTW-NL with heterogeneous recharging technologies and nonlinear recharging functions. It is a BP&C algorithm, relying on a tailored bidirectional labeling algorithm to solve the pricing problem. The labeling algorithm is capable of handling the interdependency between time and state of charge arising because of scheduling of recharge operations. The algorithmic performance is tested on a new set of benchmark instances as well as compared with well-known exact solution methods for both the E-VRPTW-L and the E-VRP-NL. Our algorithm is capable of solving instances with three different recharging technologies represented with piecewise linear functions consisting of three segments and where there are 100 customers and 21 recharging stations. The performance of the proposed method is competitive with a well-known exact solution method for the E-VRPTW-L, and a slightly modified version solves more instances than the state-of-the-art exact solution methods for the E-VRP-NL. For solving the pricing problem, a bidirectional labeling algorithm is shown to clearly outperform a monodirectional one.

Moreover, we have studied various ways of simplifying the recharging functions to see how it affects the solution quality. We have tested both linearizing the recharging functions and having homogeneous rather than heterogeneous functions. The results indicate that simplifying the recharging functions has little impact on the objective value, with all simplifications ending up within 1% of the optimal solution on average. However, the feasibility of the solutions produced by the simplified recharging functions, when evaluated on the original nonlinear functions, varies. For instances with wide time windows, the representation of the recharging functions seems to be less important, and using homogeneous linear recharging functions is sufficient. However, the results indicate that when the time windows are narrow, it is important to represent the heterogeneity of the recharging functions in the problem. Keeping the nonlinearity of the recharging functions seems to be less important as linear recharging functions obtained by taking the secant of each of the original nonlinear functions leads to feasible routes in 99% of the cases.

Interesting topics for future research might be to extend the solution method to encapsulate more realistic modeling of the energy consumption. That might involve, for example, load-dependent or uncertain energy consumption. Speed optimization might also be an interesting direction as the energy consumption is a nonlinear function of the vehicle speed.

Acknowledgments

The authors thank the anonymous reviewers for their valuable comments and suggestions, which have significantly improved the quality of this paper. Their insightful feedback has helped refine the authors' methodology, clarify their contributions, and strengthen the overall presentation. The authors also appreciate the editorial team's efforts in managing the review process.

References

Adsanver B, Coban E, Balcik B (2025) A predictive multistage post-disaster damage assessment framework for drone routing. *Internat. Trans. Oper. Res.* 32(2):626–668.

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

Baum M, Dibbelt J, Wagner D, Zündorf T (2020) Modeling and engineering constrained shortest path algorithms for battery electric vehicles. *Transportation Sci.* 54(6):1571–1600.

Baum M, Dibbelt J, Gemsa A, Wagner D, Zündorf T (2019) Shortest feasible paths with charging stops for battery electric vehicles. *Transportation Sci.* 53(6):1627–1655.

Bezzi D, Ceselli A, Righini G (2023) A route-based algorithm for the electric vehicle routing problem with multiple technologies. *Transportation Res. Part C Emerging Tech.* 157:104374.

Carey N, Steitz C (2021) EU proposes effective ban for new fossil-fuel cars from 2035. *Reuters* (July 14), https://www.reuters.com/business/retail-consumer/eu-proposes-effective-ban-new-fossil-fuel-car-sales-2035-2021-07-14/.

Cassia A, Jabali O, Malucelli F, Pascoal M (2022) The electric vehicle shortest path problem with time windows and prize collection. Ganzha M, Maciaszek L, Paprzycki M, Ślęzak D, eds. *2022 17th Conf. Comput. Sci. Intelligence Systems (FedCSIS) (Sofia, Bulgaria)*, 313–322.

Ceselli A, Felipe Á, Ortuño MT, Righini G, Tirado G (2021) A branch-and-cut-and-price algorithm for the electric vehicle routing problem with multiple technologies. *Oper. Res. Forum* 2:8.

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

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

Desaulniers G, Desrosiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D (1998) A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. Crainic TG, Laporte G, eds. *Fleet Management and Logistics* (Springer US, Boston), 57–93.

Desrochers M, Soumis F (1988) A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR Inform. Systems Oper. Res.* 26(3):191–212.

Duman EN, Taş D, Çatay B (2022) Branch-and-price-and-cut methods for the electric vehicle routing problem with time windows. *Internat. J. Production Res.* 60(17):5332–5353.

Erdelić T, Carić T (2019) A survey on the electric vehicle routing problem: Variants and solution approaches. *J. Adv. Transportation* 2019:5075671.

Erdoğan S, Miller-Hooks E (2012) A green vehicle routing problem. *Transportation Res. Part E Logist. Transportation Rev.* 48(1):100–114.

Felipe Á, Ortuño MT, Righini G, Tirado G (2014) A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Res. Part E Logist. Transportation Rev.* 71:111–128.

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.

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

Kancharla S, Ramadurai G (2020) Electric vehicle routing problem with non-linear charging and load-dependent discharging. *Expert Systems Appl.* 160:113714.

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.

Lee C (2021) An exact algorithm for the electric-vehicle routing problem with nonlinear charging time. *J. Oper. Res. Soc.* 72(7):1461–1485.

Lera-Romero G, Miranda Bront JJ, Soulignac FJ (2024) A branch-cut-and-price algorithm for the time-dependent electric vehicle routing problem with time windows. *Eur. J. Oper. Res.* 312(3):978–995.

Mohammadi F, Saif M (2023) A comprehensive overview of electric vehicle batteries market. *e-Prime Adv. Electr. Engrg. Electronics Energy* 3:100127.

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.

Omidvar A, Tavakkoli-Moghaddam R (2012) Sustainable vehicle routing: Strategies for congestion management and refueling scheduling. *2012 IEEE Internat. Energy Conf. Exhibition (ENERGYCON) (Florence, Italy)*, 1089–1094.

Pecin D, Pessoa A, Poggi M, Uchoa E (2017) Improved branch-cut-and-price for capacitated vehicle routing. *Math. Programming Comput.* 9(1):61–100.

Pelletier S, Jabali O, Laporte G, Veneroni M (2017) Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models. *Transportation Res. Part B Methodological* 103:158–187.

Preis H, Frank S, Nachtigall K (2014) Energy-optimized routing of electric vehicles in urban delivery systems. Helber S, Breitner M, Rösch D, Schön C, von der Schulenburg JM, Sibbertsen P, Steinbach M, Weber S, Wolter A, eds. *Operations Research Proceedings 2012* (Springer International Publishing, Cham, Switzerland), 583–588.

Righini G, Salani M (2006) Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optim.* 3(3):255–273.

Righini G, Salani M (2008) New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks* 51(3):155–170.

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

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

Su Y, Dupin N, Parragh SN, Puchinger J (2024) A branch-and-price algorithm for the electric autonomous dial-a-ride problem. *Transportation Res. Part B Methodological* 186:103011.

Vidal T, Laporte G, Matl P (2020) A concise guide to existing and emerging vehicle routing problem variants. *Eur. J. Oper. Res.* 286(2):401–416.