Transportation Science

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

The Electric Vehicle Routing Problem with Capacitated Charging Stations

Aurélien Froger, Ola Jabali, Jorge E. Mendoza, Gilbert Laporte

To cite this article:
Aurélien Froger, Ola Jabali, Jorge E. Mendoza, Gilbert Laporte (2022) The Electric Vehicle Routing Problem with Capacitated Charging Stations. Transportation Science 56(2):460-482. https://doi.org/10.1287/trsc.2021.1111

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

TRANSPORTATION SCIENCE
Vol. 56, No. 2, March–April 2022, pp. 460–482
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

The Electric Vehicle Routing Problem with Capacitated Charging Stations

Aurélien Froger,[a] Ola Jabali,[b] Jorge E. Mendoza,[c] Gilbert Laporte[c,d]

[a] Université de Bordeaux, Centre National de la Recherche Scientifique, Bordeaux INP, Institut de Mathématiques de Bordeaux, Unité Mixte de Recherche 5251, F-33400 Talence, France; [b] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, I-20133 Milan, Italy; [c] HEC Montréal, Montreal, Quebec H3T 2A7, Canada; [d] School of Management, University of Bath, Bath BA2 7AY, United Kingdom

Contact: aurelien.froger@math.u-bordeaux.fr, https://orcid.org/0000-0002-1814-9087 (AF); ola.jabali@polimi.it, https://orcid.org/0000-0003-0697-7448 (OJ); jorge.mendoza@hec.ca, https://orcid.org/0000-0003-2473-2655 (JEM); gilbert.laporte@cirrelt.net (GL)

Received: December 18, 2020
Revised: August 31, 2021
Accepted: October 8, 2021
Published Online in Articles in Advance: December 30, 2021

https://doi.org/10.1287/trsc.2021.1111

Abstract. Electric vehicle routing problems (E-VRPs) deal with routing a fleet of electric vehicles (EVs) to serve a set of customers while minimizing an operational criterion, for example, cost or time. The feasibility of the routes is constrained by the autonomy of the EVs, which may be recharged along the route. Much of the E-VRP research neglects the capacity of charging stations (CSs) and thus implicitly assumes that an unlimited number of EVs can be simultaneously charged at a CS. In this paper, we model and solve E-VRPs considering these capacity restrictions. In particular, we study an E-VRP with nonlinear charging functions, multiple charging technologies, en route charging, and variable charging quantities while explicitly accounting for the number of chargers available at privately managed CSs. We refer to this problem as the E-VRP with nonlinear charging functions and capacitated stations (E-VRP-NL-C). We introduce a continuous-time model formulation for the problem. We then introduce an algorithmic framework that iterates between two main components: (1) the route generator, which uses an iterated local search algorithm to build a pool of high-quality routes, and (2) the solution assembler, which applies a branch-and-cut algorithm to combine a subset of routes from the pool into a solution satisfying the capacity constraints. We compare four assembly strategies on a set of instances. We show that our algorithm effectively deals with the E-VRP-NL-C. Furthermore, considering the uncapacitated version of the E-VRP-NL-C, our solution method identifies new best-known solutions for 80 of 120 instances.

Funding: This research was partly funded by the French Agence Nationale de la Recherche through project e-VRO [Grant ANR-15-CE22-0005-01] and the Canadian Natural Sciences and Engineering Research Council [Grants 436014-2013 and 2015-06189]. This support is gratefully acknowledged.
Supplemental Material: The online appendices are available at https://doi.org/10.1287/trsc.2021.1111.

Keywords: electric vehicle routing • nonlinear charging function • synchronization constraints • mixed integer linear programming • matheuristic • iterated local search • branch-and-cut

1. Introduction

In recent years, competitive prices and technological advances have made electric vehicles (EVs) an attractive alternative to internal combustion engine-powered vehicles for logistics operations (Juan et al. 2016, Pelletier, Jabali, and Laporte 2016). To allow companies and private vehicle users to fully integrate EVs in their operations and their trips, the operations research community has turned its attention to EV route planning. The resulting literature can be divided into two streams: optimal path and vehicle routing problems. As the name suggests, the first stream focuses on problems in which the objective is to find an *optimal* path from an origin to a destination on a road network while taking into account the EV's short driving range and accommodating stops at charging stations (CSs) to recharge the vehicle's battery. These problems are out of the scope of this paper, but we refer the reader to the works of Baum et al. (2019, 2020) for a thorough literature review and state-of-the-art results for these problems. We focus on the second stream, namely, electric vehicle routing problems (E-VRPs).

E-VRPs consist of designing routes to serve a set of customers using a fleet of EVs. Because of their relatively short driving range, EVs may need to detour to CSs to replenish their battery. This is especially critical in the context of midhaul or long-haul routing (Schiffer, Stütz, and Walther 2018, Villegas et al. 2018). Therefore, key decisions in E-VRPs concern not only the assignment of the customers to the vehicles and their sequencing but also where and how much to charge the vehicles.

One of the main modeling elements in E-VRPs is the charging process of batteries. Some studies assume

that EVs are fully recharged whenever they detour to a CS. That is the case, for instance, in the green vehicle routing problem (G-VRP). The G-VRP was introduced by Erdoğan and Miller-Hooks (2012) and tackled by Koç and Karaoglan (2016), Montoya et al. (2016), Andelmin and Bartolini (2017), and Bruglieri et al. (2019), among others. One of the main assumptions in the G-VRP is that the charging time is constant, meaning that the vehicle takes a fixed amount of time to charge the battery to its full capacity, independently of the initial state of charge (SoC). Authors such as Schneider, Stenger, and Goeke (2014) and Hiermann et al. (2016) relaxed this assumption and incorporated charging times that linearly depend on the SoC on arrival at the CS. As Montoya et al. (2017) pointed out, the full charge policy may lead to unnecessary out-of-the-depot charging, which translates into expensive driver idling time and overpriced energy purchases. To overcome this drawback, several researchers have studied problem variants in which the charging time is a decision variable; notable examples include the work of Felipe et al. (2014), Desaulniers et al. (2016), Montoya et al. (2017), Keskin and Çatay (2018), and Froger et al. (2019). In the first two examples, the authors assume that the charge retrieved at a CS is a linear function of the charging time. In reality, however, the battery charging process follows a nonlinear function with respect to time (Pelletier et al. 2017). To account for this, Montoya et al. (2017), Koç et al. (2018), and Froger et al. (2019) modeled the charging process using concave piecewise linear functions, whereas Lee (2021) considered a general concave and nondecreasing charging function. In the resulting problem, known as the electric vehicle routing problem with nonlinear charging function (E-VRP-NL), the charging times no longer rely solely on the quantity of energy charged but also on the initial SoC of the EV.

In all previously discussed research, the authors assume that CSs are always available to charge a vehicle, thus implicitly assuming that CSs are *uncapacitated* and can simultaneously charge an unlimited number of EVs. In practice, however, each CS has a fixed and often small number of chargers. The intuition behind neglecting the CS capacity constraints is that accounting for the detour and charging times (or costs) while planning the routes is enough to capture the impact of the charging decisions on the cost and feasibility of solutions. Nonetheless, the long charging times (which may range from tens of minutes to several hours) and the small number of chargers typically available at CSs may generate congestion.

The nature of the congestion problem at CSs largely depends on the access options available to the fleet operator. For instance, a few networks of public (as in accessible to anyone) CSs allow for charging time reservations. In this case, congestion may be neglected, and the fleet operator only needs to make sure that routes reach the CSs within the reserved time windows. To accurately model the availability of each charger at a CS, the fleet operator may also manage their allocation to the EVs within the reserved time windows (see Bruglieri, Mancini, and Pisacane 2019a for an example). In practice, most public charging stations do not allow for reservations. In this case, the exact time at which EVs can access CSs is uncertain. Indeed, on arrival, the chargers may be busy, and the vehicle may have to wait in a queue or detour to a close by station. Keskin, Laporte, and Çatay (2019) dealt with this issue by explicitly considering expected (i.e., deterministic) time-dependent queuing times at CSs. Kullman, Goodson, and Mendoza (2021) went a step further and proposed dynamic optimization policies that constantly adapt charging and routing decisions depending on the state of CS queues. Although these authors demonstrated that their approaches can effectively handle CS access uncertainty, as Villegas et al. (2018) pointed out, most companies are unwilling to bear this risk, and thus decide to install their own private out-of-depot charging infrastructure (e.g., at satellite depots, company office branches, or exclusive-access parking spots in city centers).

Relying on a privately operated charging infrastructure only changes the nature of the congestion problem but does not solve it. To illustrate this point, we ran a feasibility test on the 120 best-known solutions (BKSs) for the E-VRP-NL reported in Montoya et al. (2017), limiting the number of chargers per CS to one, two, three, and four. According to our results, 55 of the BKSs become infeasible if there is only one charger per CS. This figure drops to 23 and 3 for the cases with two and three chargers, respectively. The only scenario where all BKSs are feasible is when CSs have four chargers (see Online Appendix A for details). Intuitively, one may think that by merely shifting the starting time of the charging operations, the feasibility problem will be solved. However, our experiments show that this is not only unnecessarily expensive (albeit with a very limited time increase), but more importantly it may be infeasible. Another option to mitigate the effects of congestion is to increase the number of chargers installed at each CS, but this may not be a viable solution in practice. Indeed, Gnann et al. (2018) predict that by the end of 2020, the purchase and installation costs of a fast charging point (i.e., power rates above 22 kW) will be around 40,000€, and its annual operation cost will reach 4,000€. Thus, if a company decides to invest in out-of-the-depot charging infrastructure, there is little chance that it will decide to install more than a couple of chargers at each CS. In conclusion, we argue that in most practical situations, the congestion at the CSs

should be taken into account when planning the routes by explicitly modeling the capacity constraints.

We are aware of two works taking into account CS capacity constraints when optimizing routing decisions in the context of a private charging infrastructure. Bruglieri, Mancini, and Pisacane (2019a) considered the G-VRP with capacitated alternative fuel stations (G-VRP-CAFS) as an extension of the G-VRP to account for the number of pumps at every refueling station. They introduced a path-based formulation and solved it using a cutting plane algorithm where the capacity constraints are separated when needed. To model the capacity constraints, the nondominated paths between fuel stations are replicated according to the number of fueling pumps at the origin and destination of the paths. The authors reported optimal solutions on instances up to 20 customers. Bruglieri, Mancini, and Pisacane (2021) tackled the same problem and proposed an improved path-based formulation that does not require path replication and solved it by a cutting plane algorithm. The capacity constraints are modeled using Helly's theorem in dimension one. Instances with up to 30 customers and four pumps at fuel stations are optimally solved. Out of the field of vehicle routing, some researchers have studied related problems. For instance, Sassi and Oulamara (2014) and Pelletier, Jabali, and Laporte (2018) considered the problem of scheduling charging operations at the depot, assuming that the routes are given as an input. Both papers considered not only constraints on the number of available chargers, but also electricity grid constraints limiting the amount of power that can be drawn from the electric grid at any given time. The latter is typically a binding constraint for central depots with a large number of available chargers but is usually not a concern for CSs because they are designed to operate at full capacity. Brandstätter, Leitner, and Ljubić (2020) studied the problem of finding optimal locations and sizes for charging stations based on the number of expected trips. They explicitly took charging station capacity into account using a time-expanded location graph.

To help the readers find their way through the different assumptions, Table 1 summarizes the main E-VRPs variants addressed in the literature (it excludes the literature on location-routing problems). The table is not exhaustive, but it provides a good overview of the current state of the research on E-VRPs.

In this paper, we introduce the E-VRP-NL with capacitated CSs (E-VRP-NL-C). The problem extends classical E-VRPs to account for the fixed number of chargers available at each CS. We assume that the CSs are privately operated. There are some key differences between the E-VRP-NL-C and the G-VRP-CAFS studied in Bruglieri, Mancini, and Pisacane (2019a) and Bruglieri, Mancini, and Pisacane (2021). The G-VRP-CAFS assumes that every EV is fully charged in constant time when it stops at a CS, whereas the quantity of energy charged during every stop at a CS is a decision variable in the E-VRP-NL-C. Moreover, because the charging functions are piecewise linear in the latter problem, the charging time of an EV depends on its initial SoC and on the quantity of energy charged.

The E-VRP-NL-C is a complex combined routing-scheduling problem belonging to the family of VRPs with synchronization constraints. More specifically, according to the taxonomy introduced by Drexl (2012), the E-VRP-NL-C belongs to the class of VRPs with *resource synchronization*, where vehicles compete to access scarce resources (i.e., the chargers at every CS). The E-VRP-NL-C shares similarities with problems where vehicles need to wait while the loading equipment is busy. Examples of problems in this class include the log truck scheduling problem (El Hachemi, Gendreau, and Rousseau 2013, Rix, Rousseau, and Pesant 2015) and routing and scheduling problems arising in public works (Grimault, Bostel, and Lehuédé 2017). The E-VRP-NL-C also relates to VRPs with intertour resource constraints. In these problems, the scarce resources are located only at the terminal vertex of the routes (i.e., the depot). An example of a problem in this category is the VRP introduced by Hempsch and Irnich (2008), where there is a limited number of ramps at the depot, and therefore only a fixed number of vehicles can be served simultaneously. The problem that is most closely related to the E-VRP-NL-C is the VRP with location congestion introduced by Lam and Van Hentenryck (2016). In this problem, each customer has a given number of requests and a limited number of resources (e.g., conveyors). When a vehicle arrives at a customer location, it must wait until at least one resource becomes available to handle the shipping. However, there exist two fundamental differences between their problem and our E-VRP-NL-C. The first is that in our problem, visiting locations with limited resources (the CSs) is needed for feasibility reasons. As a consequence, not only the number, but also the location and duration of the charging operations depend on the configuration of the routes. In other words, the *tasks* that must be synchronized are not known a priori. The second difference is that in contrast to their problem, in our E-VRP-NL-C, the task synchronization has a direct impact on the solution cost.

The contribution of this paper is twofold. First, we propose a mixed integer linear programming (MILP) formulation for the E-VRP-NL-C, showing how CS capacity constraints can be integrated into a standard E-VRP model. Our MILP formulation does not require node replication. It models the capacity constraints in a new way with respect to the modeling approach used in Bruglieri, Mancini, and Pisacane (2019a) and

Table 1. Summary of the Literature on E-VRPs

	Charging infrastructure			Charging process				
Reference	Congestion | Access	Multiple speed		Partial policy	Function	Fleet	Model(s)	Main solution method(s)
Erdoğan and Miller-Hooks (2012)	NEG				C	H	CS-Repl	Heuristics
Felipe et al. (2014)	NEG	✓		✓	L	H	CS-Repl	Local-search algorithms, SA
Schneider, Stenger, and Goeke (2014)	NEG				L	H	CS-Repl	Hybrid VNS/TS
Goeke and Schneider (2015)	NEG				L	M	CS-Repl	ALNS
Schneider, Stenger, and Hof (2015)	NEG				C/L	H	CS-Repl	AVNS
Desaulniers et al. (2016)	NEG			✓	L	H	SP	Branch-and-pricea
Hiermann et al. (2016)	NEG				L	M	CS-Repl, SP	Branch-and-price, extended ALNS
Keskin and Çatay (2016)	NEG			✓	L	H	CS-Repl	ALNS
Koç and Karaoglan (2016)	NEG				C	H	CS-Repl, CS-Path-1	B&C, SA
Montoya et al. (2016)	NEG				C	H	—	Multispace sampling heuristic
Andelmin and Bartolini (2017)	NEG				C	H	SP (CS-Path)	Two-phase exact algorithm
Leggieri and Haouari (2017)	NEG				C	H	CS-Repl, CS-Path-1	MILP solver
Montoya et al. (2017)	NEG	✓		✓	PL	H	CS-Repl	Matheuristic (ILS + SP)
Keskin and Çatay (2018)	NEG	✓		✓	L	H	CS-Repl	Matheuristic (extended ALNS)
Andelmin and Bartolini (2019)	NEG				C	H	CS-Path	Multistart local search heuristic
Bruglieri, Mancini, and Pisacane (2019b)	NEG				C	H	CS-Path-1, CS-Path-2	MILP solver
Bruglieri et al. (2019)	NEG				C	H	Customer-Path	MILP solver, Matheuristic
Bruglieri, Mancini, and Pisacane (2019a)	SCH | PUB-R/PO				C	H	CS-Repl, Customer-Path	Cutting plane method
Froger et al. (2019)	NEG	✓		✓	PL	H	CS-Repl, CS-Path	MILP solver
Hiermann et al. (2019)	NEG			✓	L	M	—	Matheuristic (GA + LNS + SP)
Keskin, Laporte, and Çatay (2019)	DWT | PUB			✓	PL	H	CS-Repl	Matheuristic (extended ALNS)
Macrina et al. (2019)	NEG	✓		✓	L	M	CS-Repl	ILS
Lee (2021)	NEG			✓	Cv	SV	CS-Repl, Customer-Path	Branch-and-price
Bruglieri, Mancini, and Pisacane (2021)	SCH | PUB-R/PO				C	H	Customer-Path	Cutting plane method
Kullman, Goodson, and Mendoza (2021)	DYN | PUB/PO	✓		✓	PL	SV	CS-Repl	Static and dynamic policies
Our research	SCH | PO	✓		✓	PL	H	CS-Path	Matheuristic (ILS + B&C)

Notes. Charging infrastructure:

- Strategy to deal with congestion at CSs and accessibility of CSs (Congestion | Access): neglected (NEG), scheduling of charging operations according to the number of available chargers (SCH), deterministic time-dependent waiting times (DWT), dynamic decision making (DYN) | public (PUB) –accessible to anyone, public with reservation (PUB-R), privately operated (PO)
- Multiple speed: each station may charge at a different speed (e.g., fast, moderate, slow)

Charging process:

- Partial policy: the quantity of energy charged is a decision
- Function: constant charging time (C), charging times linearly depending on the quantity of energy charged (L), charging times depending on the quantity of energy charged and on the initial SoC: concave piecewise linear charging function (PL), concave and nondecreasing charging function (Cv)

Fleet: homogeneous (H), mixed (M), single vehicle (SV)
Model(s): CS replication-based formulation (CS-Repl), path-based formulation in which each path corresponds to a sequence of stops at CSs between customers and/or the depot (CS-Path), CS-Path in which every path contains at most one stop (CS-Path-1) or two stops at a CS (CS-Path-2), path-based formulation in which each path corresponds to a sequence of visits to customers between CSs and/or the depot (Customer-Path), classical set partitioning with a variable per route (SP)
Methods: SA, simulated annealing; VNS, variable neighborhood search; TS, tabu search; ALNS, adaptive large neighborhood search; B&C, branch-and-cut; AVNS, adaptive VNS; ILS, iterated local search; GA, genetic algorithm.
[a]Improved results can be found in Desaulniers, Gschwind, and Irnich (2020)

Bruglieri, Mancini, and Pisacane (2021) for the GVRP-CAFS. In particular, we model the capacity constraints using sequencing variables and flow variables and constraints. Second, we introduce a framework to solve E-VRPs with CS capacity constraints in general, and the E-VRP-NL-C in particular. It is made up of two interacting components: a *route generator* and a *solution assembler*. The first component builds a set of high-quality solutions while relaxing the capacity constraints. The routes making up these solutions are stored in a pool that is sent to the assembler every few iterations. The latter combines routes from the pool in an attempt to build a solution meeting the capacity constraints. If such a solution does not exist or cannot be found within a given computing time, the assembler sends a signal to the generator to modify the search space to favor feasibility. For the particular case of the E-VRP-NL-C, we developed a route generator based on a new and efficient iterated local search (ILS) for the E-VRP-NL and a solution assembler based on branch-and-cut. More specifically, while solving a route selection problem, we discard, by means of cuts, selections of routes that lead to a violation of the capacity constraints or subset of routes for which the total time is underestimated. We present four assembly strategies allowing for different trade-offs between efficiency and effectiveness. We have carried out extensive computational experiments on a set of instances with different characteristics adapted from available benchmarks. The results demonstrate that our approach can handle the CS capacity constraints effectively and can tackle large instances with up to 320 customers. In addition, we report new BKS for 80 of the 120 instances of a well-established benchmark set for the closely related E-VRP-NL.

The remainder of this paper is organized as follows. Section 2 formally introduces the E-VRP-NL-C. Section 3 describes a MILP formulation of the problem. Section 4 presents the proposed solution method. Section 5 shows the computational results. Finally, Section 6 concludes and outlines research perspectives.

2. Problem Description

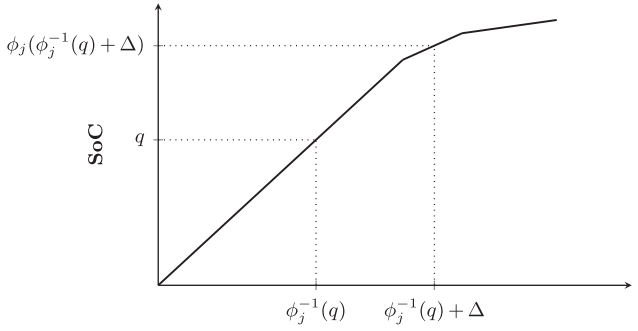
We define the E-VRP-NL-C as follows. Let I be the set of customers that need to be served and let F be the set of CSs at which recharging can take place. The CSs are privately operated, meaning that there is no uncertainty on their availability. Each customer $i \in I$ has a service time g_i. The customers are served using a homogeneous fleet of EVs. Each EV has a battery of capacity Q (expressed in Wh). At the beginning of the planning horizon, the EVs are located in a single depot, from which they leave fully charged. The depot is continuously open for T_{max} hours. Traveling from one location i (the depot, a customer, or a CS) to another location j incurs a driving time $t_{ij} \geq 0$ that corresponds to the shortest path in time from i to j and an energy consumption $e_{ij} \geq 0$ associated to this path.

Because of their limited battery capacity, EVs may need to stop en route at CSs. Charging operations can occur at any CS, they are nonpreemptive, and EVs can be partially recharged. The CS $j \in F$ has a concave piecewise linear charging function ϕ_j that maps for an empty battery the time spent charging at j and the SoC of the vehicle upon departing from j. If q is the SoC of the EV on arrival at j and Δ is the charging time, then the SoC of the EV upon departure from j is given by $\phi_j(\phi_j^{-1}(q) + \Delta)$ (Figure 1). We denote by $B_j = \{(a_{j0}, q_{j0}), \ldots, (a_{jb_j}, q_{jb_j})\}$ the totally ordered set of breakpoints of the charging function at CS j, where every breakpoint is a (charging time, SoC) pair, sorted in nondecreasing order of time.

Each CS $j \in F$ also has a *capacity*, given by the number of available chargers C_j. Because of the limited capacity of CSs, vehicles may incur waiting times while they queue for a charger. We therefore note that optimal solutions are not necessarily *left-shifted schedules*, as is the case for the E-VRP-NL and other E-VRP variants assuming uncapacitated CSs.

Feasible solutions to the E-VRP-NL-C must satisfy the following conditions: (1) each customer is visited exactly once by a single vehicle, (2) each route starts at the depot not before time 0 and ends at the depot not later than time T_{max}, (3) each route is energy feasible, that is, the SoC of an EV when it enters and leaves from any location lies between zero and Q, and (4) no more than C_j EVs simultaneously charge at each CS $j \in F$. The objective of the E-VRP-NL-C is to minimize the total time needed to serve all customers, including driving, service, charging, and waiting times. To avoid congestion at CSs, the starting times of the routes can be delayed at no cost. Moreover, an E-VRP-NL-C instance may have no feasible solution.

Figure 1. Piecewise Linear Charging Function of a CS $j \in F$



3. MILP Formulation

We can classify explicit mixed integer linear programming formulations for E-VRPs into two categories: CS replication-based formulations and path-based formulations. CS replication-based formulations are the compact MILP formulations typically used in the E-VRP literature to mathematically define the problem and solve toy instances. As their name suggests, in these formulations the nodes representing CSs are replicated and the number of stops at each CS replication is constrained to one. In this case, the number of stops at a CS is limited to its number of replications. This modeling artifice allows for tracking of the routes' travel time, distance, and SoC whenever there are several stops at the CS. To ensure that no optimal solution is cut off, the number of replications of each CS must be very large. For instance, Froger et al. (2019) provided an example where this value is equal to $4|I|$, when the congestion at each CS is neglected. CS replication-based formulations yield intractable MILPs and introduce symmetry issues.

Path-based formulations are based on a more intricate modeling strategy by which the problem is defined on a multigraph. In a nutshell, in these formulations, the nodes represent locations (the depot and the CSs or the depot and the customers) and the arcs represent possible paths between every pair of locations. A path may be defined as a sequence (possibly empty) of visits to customers between two CSs or the depot (Bruglieri et al. 2019, Bruglieri, Mancini, and Pisacane 2019a) or as a sequence of stops at CSs between two customers or the depot (Roberti and Wen 2016, Andelmin and Bartolini 2017, Froger et al. 2019). We refer to the former case a as *customer path* and to the latter as a *CS path*. In both cases, dominance rules are defined to limit the number of paths taken into consideration. These formulations no longer require CS vertices replication.

There also exist set partitioning-based formulations (Desaulniers et al. 2016), but they are out of the scope of this paper. Table 1 provides details on the formulations used in the E-VRP research.

We introduce a path-based formulation for the E-VRP-NL-C. We decided to use CS paths rather than customer paths for two reasons. First, the number of customer paths grows much faster than the number of CS paths. Second, customer path-based formulations require introducing a sufficient number of clones of paths that do not visit any customer between two CSs, whereas a CS path can at most be traveled by a single EV. We point out that the modeling of the CS capacity constraints we introduce later can easily be adapted to customer path-based formulations.

The concept of CS paths leads to a redefinition of the problem on a directed multigraph $G = (V, P)$, where $V = \{0\} \cup I$, and P is the set of arcs connecting the vertices of V. An arc in P represents a CS path p, starting in vertex $\texttt{org}(p)$ visiting a sequence of CSs or none and ending in vertex $\texttt{dest}(p)$. We denote $\texttt{n}(p)$ the number of CSs in p. If $\texttt{n}(p)$ is equal to zero, then p does not stop at any CS between $\texttt{org}(p)$ and $\texttt{dest}(p)$. Otherwise, we denote by $\texttt{cs}(p,l)$ the lth CS visited in p (with $l \in \{1,\dots,\texttt{n}(p)\}$). We define inputs $\texttt{e}(p)$ and $\texttt{t}(p)$ as the energy consumption (energy used to travel the arcs of the path) and the driving time (time to travel the arcs of the path) initially associated with CS path p. Given two vertices $i, j \in V$, we define P_{ij} as the set of CS paths connecting i to j.

Our path-based formulation of the E-VRP-NL-C involves the following decisions variables. The binary variable x_p is one if and only if an EV travels CS path $p \in P$. The continuous variables τ_p and y_p track the time and SoC of an EV when it departs from vertex $\texttt{org}(p)$ to $\texttt{dest}(p)$ using CS path p. To model the charging process at $\texttt{cs}(p,l)$, we introduce a group of variables. The continuous variables $\underline{q}_{pl}$ and $\overline{q}_{pl}$ specify the SoC of an EV when it enters and leaves $\texttt{cs}(p,l)$. For $k \in B_i \backslash \{0\}$, the binary variables $\underline{w}_{plk}$ and $\overline{w}_{plk}$ are equal to one if and only if the SoC is between $q_{\texttt{cs}(p,l),k-1}$ and $q_{\texttt{cs}(p,l),k}$ when the EV enters and leaves $\texttt{cs}(p,l)$. The variables $\underline{a}_{pl}$ and $\overline{a}_{pl}$ are the scaled arrival and departure times, according to the charging function of $\texttt{cs}(p,l)$. The continuous variables $\underline{\lambda}_{plk}$ and $\overline{\lambda}_{plk}$ represent the coefficients associated with the breakpoints $(a_{\texttt{cs}(p,l),k}, q_{\texttt{cs}(p,l),k})$ of the piecewise linear charging function, when the EV enters and leaves $\texttt{cs}(p,l)$. The continuous variable Δ_{pl} and ∇_{pl} represents the duration of the charging operation performed at $\texttt{cs}(p,l)$ and the waiting time incurred before charging at this CS. Last, the continuous variables $\underline{s}_{pl}$ and $\overline{s}_{pl}$ represent the starting and completion time of the charging operation performed at $\texttt{cs}(p,l)$.

To model the CS capacity constraints, we propose a flow-based formulation inspired from the one introduced by Artigues, Michelon, and Reusser (2003) for

the resource constrained scheduling problem. For every CS $j \in F$, we consider C_j resources (represented by the chargers). Each resource can execute at most one operation at any given time. For convenience, we introduce the set O_j of potential charging operations at CS j. Every operation $o \in O_j$ represents a stop at j at a specific position $l(o)$ in a CS path going from $\mathtt{org}(o)$ to $\mathtt{dest}(o)$. Although there may exist several CS paths between two specific vertices that stop at j at the same position, at most one these CS paths can be selected in a solution. We also introduce two dummy operations $\mathtt{o}_j^+$ and $\mathtt{o}_j^-$, which act as the source and the sink of the flow for each CS j. Specifically, the resources flow from $\mathtt{o}_j^+$ to $\mathtt{o}_j^-$ through the charging operations. Let $(o,o') \in (O_j \cup \{\mathtt{o}_j^+\}) \times (O_j \cup \{\mathtt{o}_j^-\})$ be a couple of charging operations; then the continuous flow variable $f_{oo'}$ denotes the number of chargers that are transferred from operation o to operation o'. When both these operations are not dummy, $f_{oo'}$ is equal to one if and only if these operations are scheduled on the same charger, o' is scheduled after o, and no other operation is scheduled on the charger between the completion of o and the beginning of o'. Binary variable $u_{oo'}$ is equal to one if and only if operation o' starts after the completion of operation $o \neq o'$. For notational convenience, we define $\mathtt{c}(o) := 1$ for all $o \in O_j$ and $\mathtt{c}(\mathtt{o}_j^+) := \mathtt{c}(\mathtt{o}_j^-) := C_j$.

A path-based formulation for the E-VRP-NL-C, denoted as $[P^{path}]$, is as follows:

$$[F^{path}] \ \text{minimize} \sum_{p\in P}\left(\mathtt{t}(p)x_p + \sum_{l=1}^{\mathtt{n}(p)}(\Delta_{pl}+\nabla_{pl})\right) \tag{1}$$

$$\text{subject to} \sum_{j\in V, i\neq j}\sum_{p\in P_{ij}} x_p = 1 \quad i \in I, \tag{2}$$

$$\sum_{j\in V, i\neq j}\sum_{p\in P_{ji}} x_p - \sum_{j\in V, i\neq j}\sum_{p\in P_{ij}} x_p = 0 \quad i \in V, \tag{3}$$

$$\sum_{j\in V, j\neq i}\sum_{p\in P_{ji}}\left(y_p - \mathtt{e}(p)x_p + \sum_{l=1}^{\mathtt{n}(p)}(\overline{q}_{pl} - \underline{q}_{pl})\right) = \sum_{j\in V, j\neq i}\sum_{p\in P_{ij}} y_p \quad i \in I, \tag{4}$$

$$y_p - e_{\mathtt{org}(p),\mathtt{cs}(p,1)}x_p = \underline{q}_{p1} \quad p \in P, \tag{5}$$

$$\overline{q}_{p,l-1} - e_{\mathtt{cs}(p,l-1),\mathtt{cs}(p,l)}x_p = \underline{q}_{pl} \quad p \in P, l \in \{2,\dots,\mathtt{n}(p)\}, \tag{6}$$

$$y_p - \mathtt{e}(p)x_p + \sum_{l=1}^{\mathtt{n}(p)}(\overline{q}_{pl} - \underline{q}_{pl}) \geq 0 \quad i \in I, p \in P_{i0}, \tag{7}$$

$$y_p = Qx_p \quad i \in V\backslash\{0\}, p \in P_{0i}, \tag{8}$$

$$y_p \leq Qx_p \quad p \in P, \tag{9}$$

$$\underline{q}_{pl} = \sum_{k\in B_{\mathtt{cs}(p,l)}} \underline{\lambda}_{plk} q_{\mathtt{cs}(p,l),k} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{10}$$

$$\underline{a}_{pl} = \sum_{k\in B_{\mathtt{cs}(p,l)}} \underline{\lambda}_{plk} a_{\mathtt{cs}(p,l),k} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{11}$$

$$\sum_{k\in B_{\mathtt{cs}(p,l)}} \underline{\lambda}_{plk} = \sum_{k\in B_{\mathtt{cs}(p,l)}\backslash\{0\}} \underline{w}_{plk} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{12}$$

$$\sum_{k\in B_{\mathtt{cs}(p,l)}\backslash\{0\}} \underline{w}_{plk} = x_p \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{13}$$

$$\underline{\lambda}_{pl0} \leq \underline{w}_{pl1} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{14}$$

$$\underline{\lambda}_{plk} \leq \underline{w}_{plk} + \underline{w}_{pl,k+1} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \ k \in B_{\mathtt{cs}(p,l)}\backslash\{0, b_{\mathtt{cs}(p,l)}\}, \tag{15}$$

$$\underline{\lambda}_{plb_{\mathtt{cs}(p,l)}} \leq \underline{w}_{plb_{\mathtt{cs}(p,l)}} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{16}$$

$$\overline{q}_{pl} = \sum_{k\in B_{\mathtt{cs}(p,l)}} \overline{\lambda}_{plk} q_{\mathtt{cs}(p,l)k} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{17}$$

$$\overline{a}_{pl} = \sum_{k\in B_{\mathtt{cs}(p,l)}} \overline{\lambda}_{plk} a_{\mathtt{cs}(p,l)k} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{18}$$

$$\sum_{k\in B_{\mathtt{cs}(p,l)}} \overline{\lambda}_{plk} = \sum_{k\in B_{\mathtt{cs}(p,l)}\backslash\{0\}} \overline{w}_{plk} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{19}$$

$$\sum_{k\in B_{\mathtt{cs}(p,l)}\backslash\{0\}} \overline{w}_{plk} = x_p \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{20}$$

$$\overline{\lambda}_{i0} \leq \overline{w}_{pl1} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{21}$$

$$\overline{\lambda}_{plk} \leq \overline{w}_{plk} + \overline{w}_{pl,k+1} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \ k \in B_{\mathtt{cs}(p,l)}\backslash\{0, b_{\mathtt{cs}(p,l)}\}, \tag{22}$$

$$\overline{\lambda}_{plb_{\mathtt{cs}(p,l)}} \leq \overline{w}_{plb_{\mathtt{cs}(p,l)}} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{23}$$

$$\Delta_{pl} = \overline{a}_{pl} - \underline{a}_{pl} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{24}$$

$$\sum_{j\in V, j\neq i}\sum_{p\in P_{ji}}\left(\tau_p + \mathtt{t}(p)x_p + \sum_{l=1}^{\mathtt{n}(p)}(\Delta_{pl}+\nabla_{pl})\right) + g_i = \sum_{j\in V, j\neq i}\sum_{p\in P_{ij}} \tau_p \quad i \in I, \tag{25}$$

$$\tau_p + \sum_{l=1}^{\mathtt{n}(p)}(\Delta_{pl}+\nabla_{pl}) \leq \left(T_{max} - \mathtt{t}(p) - g_{\mathtt{dest}(p)} - t_{\mathtt{dest}(p),0}\right)x_p \quad p \in P, \tag{26}$$

$$\tau_p + t_{\mathtt{org}(p),\mathtt{cs}(p,1)}x_p + \nabla_{p1} = \underline{s}_{p1} \quad p \in P, \tag{27}$$

$$\overline{s}_{p,l-1} + t_{\mathtt{cs}(p,l-1),\mathtt{cs}(p,l)}x_p + \nabla_{pl} = \underline{s}_{pl} \quad p \in P, \ l \in \{2,\dots,\mathtt{n}(p)\}, \tag{28}$$

$$\overline{s}_{pl} = \underline{s}_{pl} + \Delta_{pl} \quad p \in P, l \in \{1,\dots,\mathtt{n}(p)\}, \tag{29}$$

$$\sum_{o'\in O_j\cup\{\mathtt{o}_j^+\}} f_{o'o} = \sum_{\substack{p\in P_{\mathtt{org}(o),\mathtt{dest}(o)} \\ : \mathtt{cs}(p,l(o))=\mathtt{j}(o)}} x_p \quad j \in F, o \in O_j, \tag{30}$$

$$\sum_{o'\in O_j\cup\{\mathtt{o}_j^+\}} f_{o'o} - \sum_{o'\in O_j\cup\{\mathtt{o}_j^-\}} f_{oo'} = 0 \quad j \in F, o \in O_j, \tag{31}$$

$$\sum_{o\in O_j\cup\{\mathtt{o}_j^-\}} f_{\mathtt{o}_j^+,o} = C_j \quad j \in F, \tag{32}$$

$$\sum_{o\in O_j\cup\{\mathtt{o}_j^+\}} f_{o,\mathtt{o}_j^-} = C_j \quad j \in F, \tag{33}$$

$$\sum_{\substack{p\in P_{\mathrm{org}(o),\mathrm{dest}(o)}\\ :\mathrm{cs}(p,l(o))=\mathrm{j}(o)}} \underline{s}_{p,l(o)} - \sum_{\substack{p\in P_{\mathrm{org}(o'),\mathrm{dest}(o')}\\ :\mathrm{cs}(p,l(o'))=\mathrm{j}(o')}} \overline{s}_{p,l(o')} \geq T_{max}(u_{o'o}-1)$$

$$j\in F, o,o'\in O_j, \quad (34)$$

$$f_{oo'} \leq \min(\mathrm{C}(o),\mathrm{C}(o'))u_{oo'} \quad j\in F,(o,o')\in(O_j\cup\{\mathrm{o}_j^+\}) \times(O_j\cup\{\mathrm{o}_j^-\}), \quad (35)$$

$$x_p\in\{0,1\} \quad p\in P, \quad (36)$$

$$\tau_p\geq 0, y_p\geq 0 \quad p\in P, \quad (37)$$

$$\underline{q}_{pl},\overline{q}_{pl},\underline{a}_{pl},\overline{a}_{pl},\underline{s}_{pl},\overline{s}_{pl},\Delta_{pl},\nabla_{pl}\geq 0 \quad p\in P, l\in\{1,\dots,\mathrm{n}(p)\}, \quad (38)$$

$$\underline{\lambda}_{plk}\geq 0,\overline{\lambda}_{plk}\geq 0 \quad p\in P, l\in\{1,\dots,\mathrm{n}(p)\}, k\in B_{\mathrm{cs}(p,l)}\backslash\{0\}, \quad (39)$$

$$\underline{w}_{plk}\in\{0,1\},\overline{w}_{plk}\in\{0,1\} \quad p\in P, l\in\{1,\dots,\mathrm{n}(p)\}, k\in B_{\mathrm{cs}(p,l)}\backslash\{0\}, \quad (40)$$

$$u_{oo'}\in\{0,1\} \quad j\in F, o,o'\in O_j, \quad (41)$$

$$f_{oo'}\geq 0 \quad j\in F,(o,o')\in(O_j\cup\{\mathrm{o}_j^+\})\times(O_j\cup\{\mathrm{o}_j^-\}). \quad (42)$$

The objective function (1) minimizes the total driving, charging, and waiting time. Constraints (2) ensure that each customer is visited exactly once. Constraints (3) impose flow conservation: at each vertex the number of incoming EVs is equal to the number of outgoing EVs. Constraints (4) track the SoC of EVs at each customer location. Constraints (5) track the SoC at the arrival at the first CS of each CS path. Constraints (6) couple the SoC of an EV that leaves a CS with its SoC on arrival to the next CS of the same path. Constraints (7) ensure that if the EV travels from a customer to the depot, it has sufficient energy to reach its destination. Constraints (8) state that every EV leaves the depot with a fully charged battery. Constraints (9) couple the SoC tracking variables with the path travel variables. Constraints (10)–(24) model the relationship between the SoCs of an EV on entering and leaving a CS and the charging time. Constraints (25) track the departure time at each customer. Constraints (26) couple the time tracking variable with the path travel variables and impose the route duration limit. Constraints (27) and (28) define the starting and completion times of every charging operation, as well as the potential waiting time before the start of the operation. Constraints (29) define the completion time of the charging operations based on their starting time and duration. Constraints (30) state that a charger must be allocated to every charging operation of each selected CS path (at most one CS path in the sum can be selected in a solution). Constraints (31) ensure flow conservation for the chargers of each CS. Constraints (32) and (33) compute the flow value at the beginning and at the end of the time horizon, which is equal to the number of available chargers at each CS. Constraints (34) couple the sequencing variables with the starting time of the corresponding charging operations. Constraints (35) couple the flow variables with the operation sequence variables. Specifically, a charger can be sent from a charging operation o' to another charging operation o if o starts after the completion of o'. Finally, Constraints (36)–(42) set the domains of the decision variables.

Without preprocessing, the number of CS paths explodes with the number of CSs and the number of customers. However, a large number of these arcs cannot be part of an optimal solution. Froger et al. (2019) presented a filtering procedure to reduce the number of CS paths based on the definition of a dominance rule between two CS paths having the same origin and destination. Because of the potential waiting times that can occur at CSs, we cannot apply the dominance rule described in the work of Froger et al. (2019) between CS paths with the same origin and destination if they both contain CSs. This is because waiting times depend on all the CS paths selected to build a complete solution. In our computational experiments, we apply the dominance rule between the unique CS path with no CS between i and j (if it exists) and every other CS path of P_{ij}, with i and j in V.

4. Solution Method

In this section, we introduce a matheuristic to solve the E-VRP-NL-C. The method relies on an algorithmic framework based on two interacting components: a route generator and a solution assembler. The first component builds a pool of high-quality and diverse routes exploring the solution space of the problem without CS capacity constraints, while the second component recombines routes from the pool trying to build a solution satisfying the CS capacity constraints.

Algorithm 1 outlines the general structure of the method. The algorithm starts by generating an initial solution without taking the CS capacity constraints into account (line 1). In our implementation, this step is carried out using a modified version of the classical Clarke and Wright heuristic. Next, the algorithm enters the main loop (lines 3–27). During n_{max} iterations, the algorithm alternates between the *route generation* and *solution assembly* phases. In particular, the algorithm uses procedure `generateRoutes(·)` to retrieve a triplet $(\Phi_O,\Phi_R,\Omega^{ST})$, where Φ_O,Φ_R are sets of high-quality solutions for the *original* and *relaxed* problems (i.e., with and without CS capacity constraints), and Ω^{ST} is a set of independent, feasible, and high-quality routes. The latter will be referred to as the *short-term* pool, because it is reset before each call to the route generator. Next (lines 4–12),

the algorithm adds the routes making up the solutions in Φ_O and Φ_R to a *long-term* pool Ω^{LT} (never reset) while keeping track of the best solutions for both the original (s_O^*) and the relaxed problem (s_R^*). The intuition behind adding routes coming from potentially infeasible solutions to the original problem (i.e., solutions to the relaxed problem) to the long-term pool is to foster diversity. Indeed, these routes tend to be of high quality (in terms of the objective function) and might be recombined efficiently with others later.

The algorithm then enters the assembly phase (line 13). In the first step of this phase, the algorithm merges the short- and long-term pools Ω^{ST} and Ω^{LT}. Thus, seeking to combine the past and recent history of the search into a single set of routes that has a size that is manageable for the solution assembler. The assumption is that a new part of the search space has been explored since the last call to the assembler. The algorithm then calls procedure `assemble(·)` to retrieve a tuple (s', S'), where s' is the best solution and S' is the set of all improving solutions (with respect to s_O^*) for the original problem found during the call. If the assembler retrieves a solution, the algorithm adds its routes in S' to the long-term pool Ω^{LT} and updates the incumbent s_O^* (lines 14–18).

If after the call to the assembler, the algorithm still has not found a feasible solution (i.e., s_O^* is still equal to `NULL`), it implements two actions. First, it slightly modifies the search space to favor feasibility. In our implementation, we modify the search space by artificially reducing the opening hours of the depot (line 20). The underlying idea is that the reduction of the depot's operating hours would lead to shorter routes with more slack to accommodate a late departure from the depot or waiting times at charging stations during the assembly phase. Second, it tries to *repair* the best-known solution for the relaxed problem (i.e., s_R^*) by making it comply with the new solution space (line 21). In our implementation, if a route visiting n customers has a duration strictly greater than the new value T of the depot hours, the algorithm creates two new routes by adding a return to the depot after the $\lfloor n/2 \rfloor$th customer and reoptimizes the charging decisions within these routes. To avoid feasibility issues, the value of T is not considered when a route visits only one customer. The same procedure is performed on the newly created routes as long as the routes contain more than one customer, and their duration exceeds T. In contrast, if after the call to the assembler a feasible solution is available (i.e., s_O^* is different than `NULL`), the algorithm then reestablishes (if needed) the original solution space (i.e., resets the value of the depot opening hours to T_{max}) and moves to a new iteration.

In our implementation of the framework, we develop an iterative local search algorithm as the route generator and a branch-and-cut algorithm as the solution assembler. The remainder of this section describes these two components.

Algorithm 1 (Solution Method: General Structure)

```
/* f(s) denotes the value of the objective function for a solution s and we assume that f(NULL) = +∞
1  s_R^0 ← generateInitialSolution()
2  n ← 0, T ← T_max, Ω^LT ← ∅, s_R^* ← s_R^0, s_R ← s_R^0, s_FO^* ← NULL, s_O ← NULL
3  while n < n_max do
4      (Φ_O, Φ_R, Ω^ST) ← generateRoutes(s_R, s_O, T)  (see Algorithm 2)
5      for each s'_R ∈ Φ_R do
6          Add the routes of s'_R to Ω^LT
7          if f(s'_R) < f(s_R^*) then s_R^* ← s'_R
8      end
9      for each s'_O ∈ Φ_O do
10         Add the routes of s'_O to Ω^LT
11         if f(s'_O) < f(s_O^*) then s_O^* ← s'_O
12     end
13     (s', S') ← assemble (Ω^ST ∪ Ω^LT, s_O^*) (see Algorithm 3) /* s' = NULL if no improving solution is found
14     if s' ≠ NULL then
15         s_O^* ← s'
16         if f(s') < f(s_R^*) then s_R^* ← s'
17         for each s ∈ S' do Add the routes of s to Ω^LT
18     end
19     if s_O^* = NULL then
20         T ← max{T_min, α · T} /* T_min: minimum possible value for T, α < 1 a tuning parameter
21         s_R ← repair(s_R^*), s_O ← NULL
22     else
23         if T < T_max then T ← T_max
24         s_O ← s_O^*, s_R ← s_R^*
25     end
26     n ← n + 1
27 end
28 return s_O^*
```

4.1. Route Generator: An Iterated Local Search Algorithm for the E-VRP-NL

Our route generator is based on an iterated local search (ILS) algorithm that solves the E-VRP-NL while populating the pool of routes that is used to assemble solutions to the E-VRP-NL-C.

Introduced by Lourenço, Martin, and Stützle (2003), ILS is a metaheuristic that iteratively applies a local search phase to produce local optima, and perturbation mechanisms to escape from them. It is initialized with a solution generally provided by a constructive heuristic.

In our implementation, we combine ILS with a variable neighborhood descent (VND) search strategy for the local search phase. Algorithm 2 outlines the general structure of our method. First, the current best solution s_R^* is set equal to the initial solution s_R^0 provided as input. The algorithm then enters an iterative process. Except during the first iteration, it perturbs the current best solution s_R^* to escape from the current local optimum and potentially explore a new region of the search space (see Section 4.1.2). This produces a new start point s'_R for the VND that computes a new local optimum s_R to the E-VRP-NL (see Section 4.1.1) and also returns the best solution s_O to the E-VRP-NL-C it has encountered. We only check the CS capacity constraints after accepting a move (i.e., when building a new solution to the E-VRP-NL). If appropriate, the algorithm updates the best-known solutions s_R^* and s_O^*, as well as the sets Φ_R and Φ_O of improving solutions. It also populates a pool of routes Ω. We only add a route to Ω if it does not already contain a route visiting the same sequence of vertices. This procedure is reiterated until the targeted number of iterations δ^{max} has been reached. The optimization returns the sets of improving solutions to the E-VRP-NL and E-VRP-NL-C and the generated pool of routes Ω. To speed up the ILS algorithm, its implementation is based on the static move descriptor concept introduced by Zachariadis and Kiranoudis (2010), which prevents unnecessary reevaluations of moves and provides an efficient way of exploring neighborhoods (see Online Appendix B).

Algorithm 2 (Route Generating Procedure).

Input: a solution s_R^0 to the E-VRP-NL, a solution s_O^0 to the E-VRP-NL-C (possibly equal to `NULL`), a maximum route duration limit T

Output: a set Φ_R and a set Φ_O of improving solutions to the E-VRP-NL and to the E-VRP-NL-C, a pool of routes Ω

Procedure `generateRoutes`(s_R^0, s_O^0,T):

`/* We denote` $f(s)$ `the value of the objective function for a solution` s `and we assume` $f(\texttt{NULL}) = +\infty$

$\delta \leftarrow 0$, $\Phi_R \leftarrow \emptyset$, $\Phi_O \leftarrow \emptyset$, $\Omega \leftarrow \emptyset$

$s_R^* \leftarrow s_R^0$, $s_O^* \leftarrow s_O^0$

while $\delta < \delta^{max}$ **do**

  if $\delta = 0$ **then** $s'_R \leftarrow s_R^0$

  else $s'_R \leftarrow$ `perturb`(s_R^*,T) (see §4.1.2)

  (s_R,s_O) $\leftarrow$ `VND`(s'_R,T) (see Algorithm 1 in Online Appendix C)

  Add the routes of s_R to Ω

  if $f(s_R) < f(s_R^*)$ **then** $\Phi_R \leftarrow \Phi_R \cup \{s_R\}$, $s_R^* \leftarrow s_R$

  if $f(s_O) < f(s_O^*)$ **then** $\Phi_O \leftarrow \Phi_O \cup \{s_O\}$, $s_O^* \leftarrow s_O$

  $\delta \leftarrow \delta + 1$

end

return (Φ_R,Φ_O,Ω)

4.1.1. VND Search Phase.

The VND relies on an ordered list of local search operators. A single operator is considered at a time. If an improving move is found, the search restarts with the first operator of the list. Otherwise, it moves to the next operator. The search reaches a local optimum when the last operator fails to improve the current solution.

Our VND uses several classical VRP operators focusing on sequencing decisions. These operators are defined for solutions represented as sequences of customer visits without CSs. We use five vertex exchanges operators that work by relocating or exchanging customer visits: 1-0, 2-0, 1-1, 2-1, and 2-2 vertex exchanges. We also use the interroute and intraroute versions of 2-opt. We also define a specific operator for the E-VRP-NL, referred to as *separate*. This operator creates two routes from a single route by inserting a return to the depot after a customer visit. It may improve the cost of a solution if at least one CS is part of the split route. Indeed, creating two routes rather than one may decrease the total time when an expensive detour to a CS is avoided. We stop applying an operator as soon as it has found an improving move. We refer to Algorithm 1 in Online Appendix C for a description of the general scheme of the VND search phase.

We only consider CSs when evaluating local search moves. To make charging decisions in such a way that every route involved in a move has the lowest possible duration, we solve one (interroute moves) or more (intraroute moves) fixed route vehicle charging problems (FRVCPs). In a nutshell, this problem consists in inserting CSs into a fixed route (i.e., sequence of customers) trying to gain (energy) feasibility while minimizing the total duration of the route. To this end, we use the labeling algorithm described by Froger et al. (2019) that applies shortest path techniques similar to the one presented in Baum et al. (2019). After each call, the duration of the route is stored in a cache memory to avoid recomputing it. It is worth noting that this procedure yields a "true" evaluation of the move. Indeed, all local search-based metaheuristics for E-VRPs reported in the literature use only approximate evaluations of the moves. The reason is that, in a quest for computational efficiency, these approaches embed local search operators that either do not revise the charging decisions (where, when, and how much to charge) or solve the resulting FRVCP heuristically. This is a fundamental difference between our ILS and the one described in Montoya et al. (2017) (aside from using different local search and perturbation operators).

Because only improving solutions are accepted in the VND framework, we only solve FRVCPs for potentially improving moves. Therefore, we rely on a procedure that filters out unpromising or infeasible moves as done for example in Schiffer, Schneider, and Laporte (2018) and Hiermann et al. (2019). Such moves

are determined by establishing a lower bound on the duration of the routes resulting from a move. The value of this lower bound is computed as the sum of two terms: the minimum increase in time to detour to a CS between two successive vertices in the route, and a lower bound on the charging time. The latter is computed by dividing the sum of the energy consumption of the route and the minimum increase in energy consumption to detour to a CS between two successive vertices in the route, by the steepest slope for a segment of the piecewise linear charging functions. Similar ideas are used in shortest path algorithms for electric vehicles within the computation of a lower bound on the trip time from any vertex to the target vertex (Baum et al. 2019). This lower bound corresponds to the true duration of a route in the case where its energy consumption does not exceed Q. If the lower bound on the route duration exceeds the limit, then the route is infeasible. Otherwise, the procedure checks whether the route duration is stored in the cache memory and modifies the lower bound value accordingly. We use the lower bound on the duration of the routes to determine whether the move is strictly nonimproving. We refer to Algorithm 2 in Online Appendix C for a detailed explanation on this procedure. If a move is not discarded by the lower bound on the basis that it is infeasible or nonimproving, we solve FRVCPs for all the routes that have not been evaluated exactly, as long as the move remains feasible and potentially improving.

4.1.2. Perturbation Phase. Whenever we reach a local optimum, we perturb the current solution by removing geographically close customers and by reinserting them at different positions. First, we randomly select a customer $i \in I$. We then remove the κ closest customers to i from their respective routes, with κ randomly selected in the interval $[\min\{|I|,5\}, \max\{\min\{|I|, 5\}, \lceil\sqrt{|I|}\rceil\}]$. We set the distance between two customers i_1 and i_2 as equal to $0.5(t_{i_1,i_2} + t_{i_2,i_1} + (e_{i_1,i_2} + e_{i_2,i_1})/\rho^*)$. The value of ρ^* corresponds to the steepest slope for a segment of the piecewise linear charging functions. Finally, customers are reinserted in the solution one at a time and in a random order by applying the following rules. A removed customer cannot be reinserted in the same route from which it was removed. We evaluate the increase in time of every feasible insertion of the customer. This is done by reoptimizing the charging decisions. The probability of selecting a given feasible insertion is set inversely proportional to the time increase due to the insertion. If there exists no feasible insertion, we simply create a new route with the customer.

4.2. Solution Assembler: A Branch-and-Cut Method

The objective of the second component of the matheuristic is to assemble the best possible solution to the E-VRP-NL-C from the pool of routes Ω, obtained from the route generator component. We recall that the charging decisions made for a route are such that its total duration is minimized. The main challenge of this second component is to combine the routes in a solution that satisfies the CS capacity constraints.

Determining the best solution that can be built from selected routes in a pool is a strategy that has been successfully applied to several hardcore VRPs. Usually, this reduces to solving a set partitioning model (Alvarenga, Mateus, and De Tomi 2007, Subramanian, Uchoa, and Ochi 2013, Villegas et al. 2013, Montoya et al. 2017, Andelmin and Bartolini 2019). The strategy is used either as a postoptimization phase or as an intensification phase within a methaheuritic. An example of the latter is the matheuristic proposed by Subramanian, Uchoa, and Ochi (2013) to solve a class of VRPs. This approach has been mostly applied to problems without route coupling constraints (i.e., the feasibility of one route is independent of the feasibility of other routes). Because of the CS capacity constraints, the route coupling constraints need to be accounted for in the E-VRP-NL-C. To the best of our knowledge, only two studies have dealt with route coupling constraints in the assembly phase of a solution from a pool of routes, Morais, Mateus, and Noronha (2014) and Grangier et al. (2017), both in the context of cross-dock VRPs.

In this work, we propose assembling the solutions using a decomposition of the problem into a route selection master problem and a CS capacity management subproblem. The master problem consists in selecting a set of routes such that every customer is covered by exactly one route. The charging decisions within the selected routes (as imported from the first component) may lead to a violation of the CS capacity constraints. In such cases, the subproblem checks whether the CS capacity constraints can be met by revising some of the decisions in these routes.

We propose four versions of the subproblem, which primarily depend on the degree of allowed modifications on the routes selected by the master problem. In the first version (denoted by N: no possible revision), we do not revise the charging decisions in the selected routes and we only check whether the CS capacity constraints are satisfied. In the second version (denoted by D: delay), we allow delaying the starting time of each route (i.e., we postpone the departure of the EV from the depot) to satisfy the CS capacity constraints. In versions N and D, no increase in the total time of the solution is incurred. In the third version (denoted by DW: delay+waiting times), we allow delaying the starting time of each route, and we also allow vehicles to wait for a charger if a CS is overcrowded by EVs. In

DW, waiting times can only occur when a route stops several times for charging, because, when there is only one stop at a CS, delaying the starting time of the route is preferable, as this does not penalize the objective function. In the fourth version (denoted by DWR: delay+waiting times+revision of charging amounts), we also revise the amount of energy charged at the CSs. If a route contains at least two stops at CSs, we may decide to charge more during the first stop to avoid waiting at the second stop before charging the EV. We note that this strategy may sometimes avoid detouring to one of the CSs visited in the original route.

We designed a branch-and-cut algorithm to efficiently solve the assembling problem while exploiting the decomposition discussed previously. While solving the route selection problem, we dynamically solve the CS capacity management subproblem. More specifically, at each node of the branch-and-bound tree, we solve the subproblem for the current selection of routes. Depending on the version of the subproblem, we generate different cuts to discard infeasible route selections (see Section 4.2.3). For versions DW and DWR, we also add cuts to account for the underestimation of the increase in the total time to visit the customers.

We outline the general structure of the branch-and-cut method in Algorithm 3 (we focus on its most common used implementation based on interacting with the branch-and-bound procedure embedded into the MILP solver via callback routines). When provided as an input, we give to the solver the objective function value of a solution to the E-VRP-NL-C as a cutoff value (i.e., an upper bound on the value of the objective function of the master problem). Therefore, the solver only considers the solutions with an objective function value strictly less than this cutoff value. The solver returns the best solution assembled from a given pool of routes by the branch-and-cut algorithm. The component also returns a set Φ of improving solutions (compared with the initial solution s) computed throughout the algorithm. Whenever a selection of routes $\overline{x}$ at an integer node of the branch-and-bound tree does not lead to the introduction of a cut after solving the capacity management subproblem CMP.$(\overline{x})$, we save it along with the updated charging decisions (delays, waiting times, revision of charging amounts) to build a complete solution.

Algorithm 3 (Assembling Procedure (Focus on the Implementation))

Input: a pool Ω of routes/a solution s (possibly equal to `NULL`) to the E-VRP-NL-C

Output: the best solution s^* computed by the branch-and-cut algorithm/a set Φ of improving solutions to the E-VRP-NL-C (in comparison with s)

Procedure `assemble`(Ω,s)**:**

- Build the MILP formulation [*HC*1] or [*HC*2] of the route selection problem from pool Ω (see Sections 4.2.1 and 4.2.2)
- Define the callback function (function called at every node of the branch-and-bound tree) as procedure `SolveSubProblem`$(\overline{x})$ (see Algorithm 4) and associate to it a CPU time limit of τ^{SP} per call
- Give the model [*HC*.], the callback function, the cutoff value $f(s)$ (objective function value of s) to the MILP solver and launch it with a CPU time limit equal to τ
- Retrieve the best solution s^* computed within the CPU time limit and the set Φ of improving solutions (compared with s)

return (s^*,Φ)

In the following sections, we provide a detailed description of our four versions of the CS capacity management subproblem. We use the following notation. The binary parameter $\mathtt{a}(r,i)$ is equal to one if and only if route $r \in \Omega$ serves customer $i \in I$. We define parameter $\mathtt{t}(r)$ as the duration of a route $r \in \Omega$, obtained from the route generation component. We denote by $O(\Omega)$ the set containing all the charging operations occurring in the routes belonging to Ω, and by $O_j(\Omega) \subseteq O(\Omega)$ all the charging operations occurring at CS $j \in F$ in these routes. We denote by $O(\{r\})$ the list of charging operations occurring in route r. Let $\mathtt{r}(o)$ and $\mathtt{j}(o)$ be the route and the CS associated with a charging operation o. Let $\mathtt{S}(o)$ and $\mathtt{d}(o)$ be the starting time and duration of charging operation o in the route $\mathtt{r}(o) \in \Omega$. For each route $r \in \Omega$, we assume that the operations in $O(\{r\})$ are sorted in nondecreasing order of their starting times. We denote by $\mathtt{suc}(o)$ the charging operation following o in route $\mathtt{r}(o)$. If there does not exist any charging operation after o, we set $\mathtt{suc}(o) = -1$. For every route $r \in \Omega$ and for each charging operation $o \in O(\{r\})$ we denote by $\mathtt{t}^+(o)$ the total time (driving and customer serving) spent in r by the EV between its departure from $\mathtt{j}(o)$ and its arrival at $\mathtt{j}(\mathtt{suc}(o))$ or at the depot if $\mathtt{suc}(o) = -1$. For each operation o, we also introduce two parameters $\mathtt{ES}(o)$ and $\mathtt{LS}(o)$ representing its earliest and latest possible starting times. The values of these parameters depend on the different versions of the subproblem and they are specified in the following sections.

4.2.1. N and D Subproblem Versions. To model the route selection problem, we introduce a binary variable x_r equal to one if and only if route $r \in \Omega$ is selected. A MILP formulation of this problem is then the following classical set partitioning model:

$$[HC1] \quad \text{minimize} \sum_{r\in\Omega} \mathtt{t}(r)x_r, \tag{43}$$

$$\text{subject to} \quad \sum_{r\in\Omega} \mathtt{a}(r,i)x_r = 1 \quad i \in I, \tag{44}$$

$$x_r \in \{0,1\} \quad r \in \Omega. \tag{45}$$

Objective (43) is to select a subset of routes from Ω that minimizes the total duration. Constraints (44) ensure that each customer is visited exactly once. Constraints (45) set the domains of the decision variables.

Let $\Omega_{1^+}(\bar{x})$ denote the set of routes with at least one charging operation resulting from a feasible solution $\bar{x}$ to the above problem, that is, $\Omega_{1^+}(\bar{x}) = \{r \in \Omega : \bar{x}_r = 1 \wedge O(\{r\}) \neq \emptyset\}$. The routes without midroute charging do not need to be considered when verifying the CS capacity constraints. We also define $\Omega_{1^+}(\bar{x}, j) \subseteq \Omega_{1^+}(\bar{x})$ as the set of routes with a charging operation at $j \in F$.

4.2.1.1. Version N.

In this version of the subproblem, the CS capacity management subproblem does not revise the charging operations scheduled in the selected routes. It only checks the feasibility of the charging operations with respect to the capacity constraints. We therefore set $\mathtt{ES}(o) = \mathtt{LS}(o) = \mathtt{S}(o)$. Let $\text{CMP}_N(\bar{x})$ be this subproblem defined for the routes that belong to $\Omega_{1^+}(\bar{x})$. It can be decomposed into $|F|$ independent problems, one for every CS. To solve $\text{CMP}_N(\bar{x})$, for every CS $j \in F$ we apply a polynomial algorithm to check the existence of subsets of operations overloading the CS. Specifically, we call procedure `CheckCapacityCut`$(O_j(\Omega_{1^+}(\bar{x})), C_j)$ to check whether the capacity constraints are satisfied according to the scheduled charging operations in $O_j(\Omega_{1^+}(\bar{x}))$. We refer the reader to Algorithm 3 in Online Appendix D for the details of this procedure.

4.2.1.2. Version D.

The main drawback of version N is that it may discard promising routes. Indeed, in some cases simply delaying the starting time of some of the routes may allow their combination into a feasible solution to the problem. Note that shifting the start time of a route does not increase the objective function. In version D, we seek to resolve capacity violations by shifting starting times of the routes. Let $\text{CMP}_D(\bar{x})$ be this subproblem. In contrast to $\text{CMP}_N(\bar{x})$, $\text{CMP}_D(\bar{x})$ does not decompose into an independent problem for each CS. Let o be a charging operation. Its earliest starting time $\mathtt{ES}(o)$ is equal to $\mathtt{S}(o)$ since by construction the operations are left shifted in each route of the pool. The parameter $\mathtt{LS}(o)$ is computed by subtracting from T_{max} the time needed to complete the route (considering the duration of the next charging operations, the driving times, and no waiting times). Specifically, $\mathtt{LS}(o) = T_{max} - \sum_{o' \in O(\{\mathtt{r}(o)\}):\mathtt{S}(o') \geq \mathtt{S}(o)} (\mathtt{t}^+(o') + \mathtt{d}(o'))$.

We now define a continuous-time MILP formulation of $\text{CMP}_D(\bar{x})$. Let the variable S_o define the starting time of operation $o \in O(\Omega_{1^+}(\bar{x}))$. We model the capacity constraints using the flow-based formulation used in Section 3. For each CS $j \in F$, we consider two dummy operations $\mathtt{o}_j^+$ and $\mathtt{o}_j^-$, and we define $\mathtt{C}(o) := 1$ for all $o \in O_j(\Omega_{1^+}(\bar{x}))$ and $\mathtt{C}(\mathtt{o}_j^+) := \mathtt{C}(\mathtt{o}_j^-) := C_j$. We then introduce the continuous variable $f_{oo'}$ representing the quantity of resource (i.e., chargers) that is transferred from charging operation o to charging operation o'. We also define the sequential binary variable $u_{oo'}$ taking the value of one if operation o is processed before operation o'. The MILP formulation of $\text{CMP}_D(\bar{x})$ is as follows:

$$[\text{CMP}_D(\bar{x})] \quad \text{minimize} \quad 0, \tag{46}$$

$$\text{subject to} \quad u_{oo'} + u_{o'o} \leq 1 \quad j \in F, o, o' \in O_j(\Omega_{1^+}(\bar{x})) : o < o', \tag{47}$$

$$u_{oo''} \geq u_{oo'} + u_{o'o''} - 1 \quad j \in F, o, o', o'' \in O_j(\Omega_{1^+}(\bar{x})), \tag{48}$$

$$S_{o'} \geq S_o + \mathtt{d}(o)u_{oo'} + (\mathtt{LS}(o) - \mathtt{ES}(o'))(u_{oo'} - 1) \quad j \in F, o, o' \in O_j(\Omega_{1^+}(\bar{x})), \tag{49}$$

$$S_{\mathtt{suc}(o)} = S_o + \mathtt{d}(o) + \mathtt{t}^+(o) \quad o \in O(\Omega_{1^+}(\bar{x})) : \mathtt{suc}(o) \neq -1, \tag{50}$$

$$\sum_{o' \in O_j(\Omega_{1^+}(\bar{x})) \cup \{\mathtt{o}_j^+\}} f_{o'o} = 1 \quad j \in F, o \in O_j(\Omega_{1^+}(\bar{x})), \tag{51}$$

$$\sum_{o' \in O_j(\Omega_{1^+}(\bar{x})) \cup \{\mathtt{o}_j^+\}} f_{o'o} - \sum_{o' \in O_j(\Omega_{1^+}(\bar{x})) \cup \{\mathtt{o}_j^-\}} f_{oo'} = 0 \quad j \in F, o \in O_j(\Omega_{1^+}(\bar{x})), \tag{52}$$

$$\sum_{o \in O_j(\Omega_{1^+}(\bar{x})) \cup \{\mathtt{o}_j^-\}} f_{\mathtt{o}_j^+ o} = C_j \quad j \in F, \tag{53}$$

$$\sum_{o \in O_j(\Omega_{1^+}(\bar{x})) \cup \{\mathtt{o}_j^+\}} f_{o\mathtt{o}_j^-} = C_j \quad j \in F, \tag{54}$$

$$f_{oo'} \leq \max(\mathtt{C}(o), \mathtt{C}(o'))u_{oo'} \quad j \in F, o, o' \in O_j(\Omega_{1^+}(\bar{x})), \tag{55}$$

$$\mathtt{ES}(o) \leq S_o \leq \mathtt{LS}(o) \quad o \in O(\Omega_{1^+}(\bar{x})), \tag{56}$$

$$f_{oo'} \geq 0 \quad j \in F, (o, o') \in (O_j(\Omega_{1^+}(\bar{x})) \cup \{\mathtt{o}_j^+\}) \cup (O_j(\Omega_{1^+}(\bar{x})) \cup \{\mathtt{o}_j^-\}), \tag{57}$$

$$u_{oo'} \in \{0,1\} \quad j \in F, o, o' \in O_j(\Omega_{1^+}(\bar{x})). \tag{58}$$

The subproblem $\text{CMP}_D(\bar{x})$ is a feasibility problem. Constraints (47) state that for two distinct operations o and o', either o precedes o', o' precedes o, or o and o' are processed in parallel (if there is more than one charger at the CS). Constraints (48) express the transitivity of the precedence relations. Constraints (49) are the disjunctive constraints on the operations related to the same CS. Each such constraint is active when $u_{oo'} = 1$ and, in which case, it enforces the precedence relation between charging operations o and o'. No waiting times can occur before a charging operation. Constraints (50) enforce the precedence relation and the time lag between the charging operations occurring in the same route. Constraints (51) state that a charger must be allocated to each charging operation. Constraints (52) ensure flow conservation. Constraints

(53) and (54) define the value of the flow leaving the source and the flow entering the sink. Constraints (55) couple the flow variables with the sequence variables. Constraints (56) and (58) define the domains of the decision variables.

4.2.2. DW and DWR Subproblem Versions. In these versions of the subproblem, we allow a possible increase in the total duration of the routes. Indeed, introducing waiting times at CSs or revising the amount of charged energy may help resolve capacity violations, but such modifications may extend routes because of the nonlinearity of the charging functions and the consideration of multiple charging technologies. Let θ be a nonnegative variable estimating the added delay when solving the CS capacity management subproblem. A MILP formulation of the route selection problem (derived directly from [*HC*1]) follows:

$$[HC2] \quad \text{minimize} \sum_{r\in\Omega} \mathtt{t}(r)x_r + \theta \tag{59}$$

$$\text{s.t.} \quad (44),(45),$$

$$\theta \geq 0. \tag{60}$$

Thereafter, we assume that we have a fixed selection $\Omega_{1^+}(\bar{x})$ of routes given by fixing the variables $\{x_r\}_{r\in\Omega}$ to values respecting the current constraints of the route selection problem.

4.2.2.1. Version DW. In this version of the subproblem, we assume that EVs can wait at CSs if delaying the starting times of the routes is not sufficient to avoid capacity violations. Let $\text{CMP}_{DW}(\bar{x})$ be the scheduling subproblem of the routes $\Omega_{1^+}(\bar{x})$, which has the objective of minimizing the addition of waiting times. The MILP formulation of $\text{CMP}_{DW}(\bar{x})$ uses the decision variables S_o, $f_{oo'}$, $u_{oo'}$ defined in $\text{CMP}_D(\bar{x})$. We also introduce variable ∇_o that represents the waiting time incurred before the start of charging operation $o \in O(\Omega_{1^+}(\bar{x}))$. For every charging operation o, its earliest starting time $\mathtt{ES}(o)$ is equal to $\mathtt{S}(o)$ and its latest starting time $\mathtt{LS}(o)$ is equal to $T_{max} - \sum_{o'\in O(\{\mathtt{r}(o)\}):\mathtt{S}(o')\geq\mathtt{S}(o)}(\mathtt{t}^+(o') + \mathtt{d}(o'))$. The MILP formulation of $\text{CMP}_{DW}(\bar{x})$ is as follows:

$$[\text{CMP}_{DW}(\bar{x})] \quad \text{minimize} \sum_{o\in O(\Omega_{1^+}(\bar{x}))} \nabla_o, \tag{61}$$

$$\text{s.t.} \quad (47)\text{-}(49),(51)\text{-}(58),$$

$$S_{\mathtt{suc}(o)} = S_o + \mathtt{d}(o) + \mathtt{t}^+(o) + \nabla_{\mathtt{suc}(o)} \quad o \in O(\Omega_{1^+}(\bar{x})) : \mathtt{suc}(o) \neq -1, \tag{62}$$

$$\nabla_o \geq 0, \quad o \in O(\Omega_{1^+}(\bar{x})). \tag{63}$$

Objective (61) is to minimize the waiting time inserted in each route. Constraints (62) define the minimum time lag between the charging operations occurring in the same route. Constraints (63) define the domains of the waiting decision variables.

4.2.2.2. Version DWR. In this version of the subproblem, in addition to the introduction of waiting times, resolving capacity violations at CSs can also be achieved by revising the amounts of energy charged at each CS in every route. We denote by $\text{CMP}_{DWR}(\bar{x})$ the subproblem in which we want to minimize the increase in the duration of the selected routes. Indeed, revising the charging operations leads to an increase in time when the CSs in a route have different charging technologies or when the charging functions are nonlinear. If we substantially increase the charging amounts at a CS, we may not need to stop at the subsequent CS in the route. We therefore need to account for the potential removal of stops at CSs.

We denote by $\mathtt{e}^+(o)$ the energy consumption of the EV between its departure from $\mathtt{j}(o)$ and its arrival at $\mathtt{j}(\mathtt{suc}(o))$ if $\mathtt{suc}(o) \neq -1$. This takes into account the energy consumed to visit all the customers scheduled in the route between charging operations o and $\mathtt{suc}(o)$ or the depot. Because a charging operation can be skipped by charging more energy during the previous or next charging operations of the same route, we define $\widetilde{\mathtt{t}}(o)$ and $\widetilde{\mathtt{e}}(o)$ as the driving time and energy saved if the EV does not detour to perform the charging operation o. We define $\Omega_{2^+}(\bar{x})$ as the subset of $\Omega_{1^+}(\bar{x})$ that contains only the routes including at least two charging operations (i.e., $\Omega_{2^+}(\bar{x}) = \{r \in \Omega_{1^+}(\bar{x}) : |O(\{r\})| \geq 2\}$).

Our formulation of $\text{CMP}_{DWR}(\bar{x})$ draws on formulation $[\text{CMP}_{DW}(\bar{x})]$. Aside from using decision variables defined in the latter, $[\text{CMP}_{DWR}(\bar{x})]$ also uses the following decision variables for the operations in $O(\Omega_{2^+}(\bar{x}))$. The variables $\underline{t}_o$ and $\bar{t}_o$ are the scaled arrival and departure times, according to the charging function of CS $\mathtt{j}(o)$. The binary variables $\underline{w}_{ok}$ and $\overline{w}_{ok}$ are equal to one if and only if the SoC lies between $q_{\mathtt{j}(o),k-1}$ and $q_{\mathtt{j}(o),k}$, with $k \in B_{\mathtt{j}(o)}\backslash\{0\}$, on starting and finishing operation o, respectively. The continuous variables $\underline{\lambda}_{ok}$ and $\overline{\lambda}_{ok}$ are the coefficients associated with the breakpoints $(a_{\mathtt{j}(o),k}, q_{\mathtt{j}(o),k})$ of $\phi_{\mathtt{j}(o)}$ upon starting and finishing operation o, respectively. The continuous variables $\underline{y}_o$ and $\overline{y}_o$ represent the SoC of the EV on starting and finishing charging operation o. The continuous variable Δ_o represents the duration of charging operation o.

For each route, we check whether it might be possible for a charging operation o to be skipped by considering that the EV leaves the previous CS (or depot) with a fully replenished battery. If this allows the EV to reach the next CS or to return to the depot without performing o, then we allow the EV not to detour to the corresponding CS. To this end, we introduce the binary variable z_o equal to one if and only if the charging operation o is executed.

We also compute for every charging operation the time windows during which it must be scheduled. The earliest starting time $\mathtt{ES}(o)$ of a charging operation o is computed assuming that the EV skips (if the previous computation has shown it is possible) the previous charging operations (if any), and charges the maximum between the energy needed to recover the detour to the CS and the energy required to reach the next CS. To compute the latter, we consider that the SoC of the EV on arriving at the CS is maximal (i.e., a full charge occurs at the previous CS). Then, we estimate the charging times assuming that the EV arrives with an empty battery. The latest starting time $\mathtt{LS}(o)$ of operation o is computed assuming that the EV returns to the depot at time T_{max} and assuming that the EV skips the next charging operations (if possible).

The MILP formulation of $\mathrm{CMP}_{DWR}(\overline{x})$ is as follows:

$$[\mathrm{CMP}_{DWR}(\overline{x})] \text{ minimize } \sum_{o\in O(\Omega_{1^+}(\overline{x}))} \nabla_o + \sum_{o\in O(\Omega_{2^+}(\overline{x}))} (\Delta_o - \mathtt{d}(o) - (1-z_o)\widetilde{\mathtt{t}}(o)) \tag{64}$$

$$\text{s.t. (47)-(48), (52)-(58), (63),}$$

$$\underline{y}_o = \sum_{k\in B_{\mathtt{j}(o)}} \underline{\lambda}_{ok} q_{\mathtt{j}(o)k} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{65}$$

$$\underline{t}_o = \sum_{k\in B_{\mathtt{j}(o)}} \underline{\lambda}_{ok} a_{\mathtt{j}(o)k} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{66}$$

$$\sum_{k\in B_{\mathtt{j}(o)}} \underline{\lambda}_{ok} = \sum_{k\in B_{\mathtt{j}(o)}\setminus\{0\}} \underline{w}_{ok} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{67}$$

$$\sum_{k\in B_{\mathtt{j}(o)}\setminus\{0\}} \underline{w}_{ok} = z_o \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{68}$$

$$\underline{\lambda}_{o0} \le \underline{w}_{o1} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{69}$$

$$\underline{\lambda}_{ok} \le \underline{w}_{ok} + \underline{w}_{o,k+1} \quad o \in O(\Omega_{2^+}(\overline{x})), k \in B_{\mathtt{j}(o)}\setminus\{0, b_{\mathtt{j}(o)}\}, \tag{70}$$

$$\underline{\lambda}_{ob_{\mathtt{j}(o)}} \le \underline{w}_{ob_{\mathtt{j}(o)}} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{71}$$

$$\overline{y}_o = \sum_{k\in B_{\mathtt{j}(o)}} \overline{\lambda}_o q_{\mathtt{j}(o)k} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{72}$$

$$\overline{t}_o = \sum_{k\in B_{\mathtt{j}(o)}} \overline{\lambda}_o a_{\mathtt{j}(o)k} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{73}$$

$$\sum_{k\in B_{\mathtt{j}(o)}} \lambda_{ok} = \sum_{k\in B_{\mathtt{j}(o)}\setminus\{0\}} \overline{w}_{ok} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{74}$$

$$\sum_{k\in B_{\mathtt{j}(o)}\setminus\{0\}} \overline{w}_{ok} = 1 \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{75}$$

$$\overline{\lambda}_{o0} \le \overline{w}_{o1} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{76}$$

$$\overline{\lambda}_{ok} \le \overline{w}_{ok} + \overline{w}_{o,k+1} \quad o \in O(\Omega_{2^+}(\overline{x})), k \in B_{\mathtt{j}(o)}\setminus\{0, b_{\mathtt{j}(o)}\}, \tag{77}$$

$$\overline{\lambda}_{ob_{\mathtt{j}(o)}} \le \overline{w}_{ob_{\mathtt{j}(o)}} \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{78}$$

$$\Delta_o = \overline{t}_o - \underline{t}_o \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{79}$$

$$\Delta_o \le a_{\mathtt{j}(o),b_{\mathtt{j}(o)}} z_o \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{80}$$

$$\underline{y}_{\mathtt{o}_{\mathtt{first}}(r)} = \mathtt{q}_{\mathtt{first}}(r) \quad r \in \Omega_{2^+}(\overline{x}), \tag{81}$$

$$\underline{y}_{\mathtt{suc}(o)} = \overline{y}_o - \mathtt{e}^+(o) + \widetilde{\mathtt{e}}(o)(1-z_o) \quad o \in O(\Omega_{2^+}(\overline{x})) : \mathtt{suc}(o) \ne -1, \tag{82}$$

$$\overline{y}_o - \mathtt{e}^+(o) + \widetilde{\mathtt{e}}(o)(1-z_o) \ge 0 \quad o \in O(\Omega_{2^+}(\overline{x})) : \mathtt{suc}(o) = -1, \tag{83}$$

$$S_{o'} \ge S_o + \mathtt{d}(o)u_{oo'} + (\mathtt{LS}(o) - \mathtt{ES}(o'))(u_{oo'} - 1) \quad j \in F, o, o' \in O_j(\Omega_{1^+}(\overline{x}) \setminus \Omega_{2^+}(\overline{x})), \tag{84}$$

$$S_{o'} \ge S_o + \Delta_o + (\mathtt{LS}(o) - \mathtt{ES}(o'))(u_{oo'} - 1) \quad j \in F, o, o' \in O_j(\Omega_{2^+}(\overline{x})), \tag{85}$$

$$S_{\mathtt{suc}(o)} = S_o + \mathtt{d}(o) + \mathtt{t}^+(o) + \nabla_{\mathtt{suc}(o)} \quad o \in O(\Omega_{1^+}(\overline{x})\setminus\Omega_{2^+}(\overline{x})) : \mathtt{suc}(o) \ne -1, \tag{86}$$

$$S_{\mathtt{suc}(o)} = S_o + \Delta_o + \mathtt{t}^+(o) - \widetilde{\mathtt{t}}(o)(1-z_o) + \nabla_{\mathtt{suc}(o)} \quad o \in O(\Omega_{2^+}(\overline{x})) : \mathtt{suc}(o) \ne -1, \tag{87}$$

$$S_o + \Delta_o + \mathtt{t}^+(o) - \widetilde{\mathtt{t}}(o)(1-z_o) \le T_{max} \quad o \in O(\Omega_{2^+}(\overline{x})) : \mathtt{suc}(o) = -1, \tag{88}$$

$$\sum_{o'\in O_{\mathtt{j}(o)}(\Omega_{1^+}(\overline{x}))\cup\{\mathtt{o}^+_{\mathtt{j}(o)}\}} f_{o'o} = 1 \quad o \in O(\Omega_{1^+}(\overline{x})\setminus\Omega_{2^+}(\overline{x})), \tag{89}$$

$$\sum_{o'\in O_{\mathtt{j}(o)}(\Omega_{1^+}(\overline{x}))\cup\{\mathtt{o}^+_{\mathtt{j}(o)}\}} f_{o'o} = z_o \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{90}$$

$$z_o \in \{0,1\}, \Delta_o \ge 0, 0 \le \underline{y}_o \le Q, 0 \le \overline{y}_o \le Q \quad o \in O(\Omega_{2^+}(\overline{x})), \tag{91}$$

$$\underline{w}_{ok}, \overline{w}_{ok} \in \{0,1\}, \quad o \in O(\Omega_{2^+}(\overline{x})), k \in B_{\mathtt{j}(o)}\setminus\{0\}, \tag{92}$$

$$\underline{\lambda}_{ok}, \overline{\lambda}_{ok} \in \{0,1\} \quad o \in O(\Omega_{2^+}(\overline{x})), k \in B_{\mathtt{j}(o)}. \tag{93}$$

Objective (64) is to minimize the total additional time inserted in each route. Constraints (65)–(79) model the piecewise linear charging functions. Constraints (80) impose a duration equal to zero for each charging operation that is not executed anymore. For each route $r \in \Omega_{2^+}(\overline{x})$, Constraints (81) define the SoC $\mathtt{q}_{\mathtt{first}}(r)$ of the EV on starting its first charging operation (denoted $\mathtt{o}_{\mathtt{first}}(r)$). Constraints (82) couple the SoC of the EV after finishing a charging operation with its SoC when starting the next charging operation occurring in the route. If z_o is equal to zero, then the SoC $\underline{y}_o = \overline{y}_o$ still takes into account the energy consumed to detour to CS $\mathtt{j}(o)$. The energy saved by not stopping at this CS is subtracted when computing the SoC at the beginning of the next operation of the route or at the arrival at the depot (see (83)). For each route, Constraints (83) force the corresponding EV to have enough SoC at the end of the last charging operation to reach the depot. Constraints (84) and (85) are the disjunctive constraints on the operations related to the same CS. Constraints (86) and (87) define a minimum time lag between the charging operations occurring in the same route. If z_o is equal to zero, then the starting time S_o still takes into account the detour to CS $\mathtt{j}(o)$. The time saved by not stopping at this CS is subtracted during the computation of the departure time

for the next operation of the route. Constraints (88) limit the route duration. Constraints (89) and (90) assign a charger to each operation that is executed. Constraints (91)–(93) define the domains of the new decision variables.

4.2.3. Cut Generation Procedure. To couple the route selection problem with the CS capacity management subproblem, we generate constraints to cut off infeasible selections of routes in $\Omega_{1^+}(\overline{x})$ and to bound the variable θ (for versions DW and DWR). The efficiency of our branch-and-cut method is based on the strength of these cuts. Rather than only cutting the current solution, we investigate a strategy that generates cuts for a large portion of the solution space of the route selection problem. Moreover, we try to add several cuts at a time to speed up the convergence of the algorithm.

We first focus on the cuts that are applicable only to version N. Given a CS j, let $\mathcal{O}_j$ be a set containing subsets of operations with a cardinality strictly larger than C_j, which overlap in time. To discard the current solution $\overline{x}$ in the route selection problem, we add to [*HC*1] the following cuts:

$$\sum_{r\in\Omega_{1^+}(\overline{x}):O(\{r\})\cap U\neq\emptyset} x_r \leq C_j \quad j\in F, U\in\mathcal{O}_j. \tag{94}$$

For the other versions of the subproblem, Algorithm 4 summarizes the procedure used to solve the capacity management subproblem and to generate cuts to discard infeasible selection of routes. We first try to detect, before solving $\mathrm{CMP}_{\cdot}(\overline{x})$, the potential infeasibility of the subproblem using two fast procedures focusing independently on each CS. If the subproblem is deemed infeasible, then we generate cuts specially crafted for our problem. If not, we solve the MILP formulation $[\mathrm{CMP}_{\cdot}(\overline{x})]$ with a commercial solver. If possible, we decompose the subproblem into smaller independent problems beforehand. We present below the details of the cut generation procedure.

The two algorithms we apply to detect capacity violations that cannot be solved by the CS capacity management subproblem are frequently used in scheduling problems. They focus on a single CS and are based on a reasoning rooted in the time windows (derived from the opening hours of the depot) in which every charging operation can be scheduled. The first algorithm focuses on the fixed part of the charging operations (line 4). We call a fixed part of a charging operation o the time interval (if it exists) between the latest starting time $\mathtt{LS}(o)$ and earliest completion time $\mathtt{ES}(o)+\mathtt{d}(o)$. Based on the fixed part of the charging operations, we detect subsets of operations that will necessarily lead to a violation of the capacity constraints, and we add cuts to forbid the underlying subset of routes (line 4). Hereafter, we refer to this algorithm as the *fixed part-based algorithm*. The details of this procedure are provided in Algorithm 8 (Online Appendix D). The second algorithm is based on an energy reasoning (line 4). The required energy consumption of a charging operation o during a time interval $[t_1,t_2)$ is equal to the minimum duration (possibly equal to zero) for which o is surely executed within the interval. For a CS j, the total required energy consumption by the charging operation o scheduled at j over time interval $[t_1,t_2)$ cannot exceed $C_j(t_2-t_1)$. The difference between the last term and the total required energy consumption is called the slack over $[t_1,t_2)$. The slack over any time interval must always be nonnegative. The number of intervals that needs consideration is bounded (see Baptiste, Le Pape, and Nuijten 2001 for more details). When there exists an interval for which a slack is strictly negative, the subset of operations having a nonzero energy consumption on this interval cannot be performed without leading to a violation of the capacity constraints. We add a no-good cut to discard it (line 4). Hereafter, we refer to this algorithm as the *energy reasoning-based algorithm*. The details of this procedure are provided in Algorithm 9 (Online Appendix D). To limit the computation time, if the two previous algorithms prove the infeasibility of the subproblem, we add to the route selection problem the generated cuts and we do not solve the subproblem.

The decomposition of the subproblem $\mathrm{CMP}_D(\overline{x})$, $\mathrm{CMP}_{DW}(\overline{x})$, or $\mathrm{CMP}_{DWR}(\overline{x})$ into several independents smaller subproblems is based on reasoning about the stops at the CSs in the different routes. The interest is to potentially formulate cuts for each of these small subproblems. Let $G(\overline{x})$ be a graph in which each vertex represents a CS and there exists an edge between two CSs if there exists a route in $\Omega_{1^+}(\overline{x})$ with charging operations at these two CSs. We can decompose the subproblem into as many independent subproblems as the number of connected components of $G(\overline{x})$. Let $\mathcal{C}(G(\overline{x}))$ be the connected components of $G(\overline{x})$. For every connected component $C\in\mathcal{C}(G(\overline{x}))$, only the charging operations scheduled at the CSs in C need consideration. We denote by $\mathrm{CMP}(\overline{x},C)$ the subproblem restricted to these operations. When $\mathrm{CMP}(\overline{x},C)$ is infeasible, we generate an integer Benders cut, also called combinatorial Benders cut (Codato and Fischetti 2006), to invalidate the current solution to the route selection problem (line 4). For versions DW and DWR of the subproblem, we also compare the current value $\overline{\theta}$ of θ with the increase in time computed over the set $\mathcal{C}^{\mathrm{feas}}$ of connected components for which the corresponding subproblem is feasible. For each subset $\mathcal{C}$ of $\mathcal{C}^{\mathrm{feas}}$, we generate an integer Benders optimality cut if the route selection problem underestimates the increase in time needed to resolve the capacity violations at CSs (line 4).

Algorithm 4 (Solving the CS Capacity Management Subproblem and Generating Cuts for Version D, DW, or DWR)

Input: a solution $\overline{x}$ to the current master problem

Procedure `SolveSubProblem`($\overline{x}$)**:**
  Compute `ES`(o) and `LS`(o) for each charging operation $o \in O(\Omega_{1^+}(\overline{x}))$
  for $j \in F$ **do**
    $\mathcal{O} \leftarrow$ `FixedPartAlgorithm`$(O_j(\overline{x}), C_j)$ (see Algorithm 4 in Online Appendix D)
    if $\mathcal{O} \neq \emptyset$ **then**
      for $U \in \mathcal{O}$ **do** Generate the following cut and add it to [*HC.*]: $\sum_{r \in \Omega_{1^+}(\overline{x},j):O(\{r\})\cap U\neq\emptyset} x_r \leq C_j$
    else
      $\mathcal{O} \leftarrow$ `EnergyAlgorithm`$(O_j(\overline{x}), C_j)$ (Algorithm 5 in Online Appendix D)
      if $\mathcal{O} \neq \emptyset$ **then**
        for $U \in \mathcal{O}$ **do** Generate the following cut and add it to [*HC.*]: $1 + \sum_{r \in \Omega_{1^+}(\overline{x},j):O(\{r\})\cap U\neq\emptyset}(x_r - 1) \leq 0$
      end
    end
  end
  if *no cuts have been generated and* $\overline{x}$ is integer **then**
    $\mathcal{C}^{\text{feas}} \leftarrow \emptyset$
    for each $C \in \mathcal{C}(G(\overline{x}))$ **do**
      Solve the MILP formulation of CMP $(\overline{x}, C)$ that corresponds to the selected version of the subproblem
      if *no solution is found* **then**
        Generate the following cut and add it to [*HC.*]: $1 + \sum_{r \in (\cup_{j\in C}\Omega_{1^+}(\overline{x},j))}(x_r - 1) \leq 0$
      else
        $\mathcal{C}^{\text{feas}} \leftarrow \mathcal{C}^{\text{feas}} \cup \{C\}$ `/* Only for versions DW and DWR`
      end
    end
    `/* Only for versions DW and DWR`
    for *every subset* $\mathcal{C}$ of connected components of $\mathcal{C}^{\text{feas}}$ **do**
      Let z(CMP($\overline{x}$,C))denote the optimal value of subproblem CMP($\overline{x}$, C)
      if $\sum_{C\in\mathcal{C}} z(\text{CMP}(\overline{x}, C)) > \overline{\theta}$ **then** Generate the following cut and add it to [*HC.*]: $(\sum_{C\in\mathcal{C}} z(\text{CMP}(\overline{x}, C)))(1 + \sum_{C\in\mathcal{C}} \sum_{r \in (\cup_{j\in C}\Omega_{1^+}(\overline{x},j))}(x_r - 1)) \leq \theta$
    end
  end
  if *no cuts have been generated* **then**
    Save the solution result of calling the subproblem on the routes of $\Omega_{1^+}(\overline{x})$
  end

5. Computational Results

We considered the 120-instance testbed of Montoya et al. (2017) (available from http://www.vrp-rep.org/). In this testbed, there are six sets of 20 instances, each with 10, 20, 40, 80, 160, or 320 customers. The EVs are Peugeot iOns that have a consumption rate of 0.125 kWh/km and a battery of 16 kWh. In these instances, the triangular inequalities hold for energy consumption. However, this assumption is not necessary for our model and solution method. When dealing with the E-VRP-NL-C, we adapted each instance by fixing a number of chargers for each CS. We decided to consider instances in which all the CSs have one or two chargers.

The first aim of our computational experiments is to assess the performance of a commercial MIP solver for solving small-size instances of the E-VRP-NL-C using the path-based formulation introduced in Section 3 (although the primary motivation for introducing the latter formulation was to give a precise definition of the problem). These results are presented in Section 5.1. The second aim of our computational experiments is to assess the quality of the ILS presented in Section 4.1 as a solution method for the E-VRP-NL. A comparison with results from the literature is presented in Section 5.2. The third aim of our experiments is to assess the effectiveness of our algorithmic framework to build high-quality solutions to the E-VRP-NL-C. We compare the results given by our matheuristic according to the version of the subproblem selected during the assembly phase. These results are presented in Section 5.3.

We implemented all the algorithms using Java 8. In all the tables, the CPU time is given in seconds and rounded to the nearest integer.

5.1. Results for the E-VRP-NL-C Formulation

We considered the twenty 10-customer and the twenty 20-customer instances. We ran the path-based model with a three-hour CPU time limit. We used Gurobi 8.1.1 through its Java API. The results were obtained using a standard PC with an Intel(R) Core(TM) i7-9700K processor clocked at 3.6 GHz, equipped with 128 GB RAM, and running Debian GNU/Linux 10. Each instance was executed on a single thread.

Table 2 reports the number of instances with a solution proven to be optimal (#Opt), the average CPU time in seconds (Time) for these latter instances, and the average gap (Gap) for the remaining instances. We compute the gap as $(z - z^{LB})/z$, where z is the objective function value of the best integer solution returned by the solver, and z_{LB} is the best lower bound retrieved by the solver. The detailed results for all the tested instances are reported in Online Appendix E.1.

Table 2. Computational Results for the CS Path-Based Formulation on the 10-Customer and 20-Customer Instances

	Capacity = 1			Capacity = 2		
$\|I\|$	#Opt	Time	Gap	#Opt	Time	Gap
10	17	873	22%	17	776	19%
20	0	—	32%[a]	0	—	29%[a]

[a]There are 16 of 20 instances for which the solver reaches the CPU time limit without finding any solution. They are not considered when computing the gap.

Our results show that the path-based model cannot solve all the 10-customer instances to optimality within the CPU time limit, whereas the results in Froger et al. (2019) showed that they can be solved within an average computation time of roughly four minutes when CS capacity constraints are ignored. We also observe that the 20-customer instances are already very challenging for the model because zero instances were solved. As expected, because the linear relaxation of the model is weak, we conclude that solving the MILP formulations with a commercial solver does not constitute an efficient solution method for the E-VRP-NL-C. However, 17 of the 40 tested instances were optimally solved when considering one or two chargers per CS (the number of chargers appears to have no impact).

5.2. Results for the E-VRP-NL

Because the route generator of our matheuristic essentially solves the E-VRP-NL, we wanted to assess its quality on this problem. We considered the 120 instances of the original Montoya et al. (2017) testbed. Each instance was executed on a single thread with 12 GB RAM and on a cluster of 27 computers, each having 12 cores and two Intel(R) Xeon X5675 3.07-GHz processors. We performed 10 test runs for each instance using Algorithm 2 and compared the results with those obtained in the E-VRP-NL literature through two solutions methods: the metaheuristic of Montoya et al. (2017), which combines an ILS with a heuristic concentration (ILS+HC), and the large neighborhood search (LNS) of Koç et al. (2018). Table 3 shows the results of this comparison. Given a number of customers in the instances, it reports for each solution method the number of BKS to the E-VRP-NL (#BKS), the average gap to the BKS (Gap), and the average number of routes in the best computed solution (#Routes). We compute the gap as $(z - z^*)/z$, where z is the objective function value of the best solution returned by the solution method, and z^* is the objective function value of the BKS. Table 3 also reports the average gap to the best average objective function value (Gap BA) as $(z_{\text{avg}} - z^*_{\text{avg}})/z_{\text{avg}}$, where z_{avg} is the average objective function value of the solutions obtained in 10 runs by the solution method, and z^*_{avg} is the best average (BA) objective function value obtained in 10 runs by one of the existing solution method for the E-VRP-NL. The detailed results for all the tested instances are reported in Online Appendix E.2.

The results presented in Table 3 clearly show that our ILS outperforms all existing methods for the E-VRP-NL. We identified 80 new BKS and matched the existing BKS for the remaining instances. We improved the previous solutions by about 4.0%. The improvement increases with the number of customers. The algorithm is also stable as the average results on 10 test runs for each instance are better than those reported for previous methods. Although it is difficult to draw definitive conclusions on the computation times because these tests were run on different machines, it seems

Table 3. Comparison of Results Obtained by ILS with Results of Montoya et al. (2017) and Koç et al. (2018) for the E-VRP-NL

	ILS + HC					LNS					ILS				
	2.33-GHz processor; 16 GB of RAM					3.6-GHz processor; 32 GB of RAM					3.07-GHz processor; 12 GB of RAM				
$\|I\|$	#BKS	Gap BKS	#Routes	Gap BA	Time	#BKS	Gap BKS	#Routes	Gap BA	Time	#BKS	Gap BKS	#Routes	Gap BA	Time
10	20	0.0%	2.7	0.3%	6	20	0.0%	2.7	0.1%	8	20	0.0%	2.7	0.0%	5
20	11	0.3%	3.7	0.7%	11	12	0.2%	3.6	0.4%	14	20	0.0%	3.6	0.0%	8
40	3	1.0%	6.5	2.6%	35	6	0.9%	6.4	1.1%	45	20	0.0%	6.2	0.0%	20
80	0	3.8%	9.2	5.4%	80	0	3.8%	8.8	3.8%	99	20	0.0%	8.6	0.0%	64
160	0	7.3%	16.7	8.1%	568	0	7.7%	16.6	7.8%	632	20	0.0%	15.3	0.0%	295
320	0	11.2%	32.6	12.6%	4398	0	11.7%	32.6	12.4%	4555	20	0.0%	29.0	0.0%	1118
All	34	3.9%	11.9	4.9%	850	38	4.1%	11.8	4.3%	892	120	0.0%	10.9	0.0%	252

Note. ILS + HC (Montoya et al. 2017); LNS (Koç et al. 2018).

Table 4. Value of the Matheuristic Parameters

n_{max}	α	T_{min}	δ^{max}	τ	τ^{SP}
12	0.9	$0.67 \cdot T_{max}$	200	180 s	5 s

that the ILS is at least as fast as the other methods, if not the fastest one. The major difference between our method and the existing ones comes from the reoptimization of the charging decisions when evaluating a move and the use of larger neighborhoods.

5.3. Results for the E-VRP-NL-C

We performed several tests to assess the effectiveness of our matheuristic in obtaining high-quality solutions for the E-VRP-NL-C. Each instance was executed on a single thread with 12 GB RAM and on a cluster of 27 computers, each having 12 cores and two Intel(R) Xeon®X5675 3.07-GHz processors. We used Gurobi 7.5.0 (through its Java API) to solve the MILP models.

After some preliminary experiments, we set the values of the parameters of the matheuristic as presented in Table 4. Setting the stopping criterion to 12 iterations and the number of ILS iterations to 200 proved to be a good compromise between solution quality and computation time. In some rare cases we were not able to optimally solve the assembly phase using version DWR of the subproblem. Indeed, when it is difficult to find a solution satisfying the CS capacity constraints or when such a solution does not exist, the MILP formulation becomes computationally expensive. Nonetheless, after testing higher CPU time limits for the second component, we observed that the impact on the results was negligible.

We first compare the results obtained by our matheuristic for the four versions of the subproblem used by the solution assembler. For one or two chargers at each CS and each version of the subproblem, we performed 10 test runs for each instance. Table 5 reports the best results obtained during our tests. Specifically, given a number of chargers per CS and a selected version of the subproblem used in the branch-and-cut algorithm, this table reports the number of instances with a solution in each of the 10 tests (#F), the number of instances with the best solution (BS) to the E-VRP-NL-C computed all over our tests (#BS), including those of the path-based formulation, and the average gap to the BS (Gap BS). Table 6 reports the average CPU time in seconds over 10 runs for the whole algorithm (T), and for the first (T_1) and second (T_2) components. It also reports the average gap with respect to the best average objective function value obtained using a given version of the subproblem (Gap BA). The gaps are computed as in Section 5.2. Detailed results for all the tested instances are reported in Online Appendix E.3.

First, it should be noted that our matheuristic returns an optimal solution for the 17 small-size instances for which the MILP path formulation yields an optimal solution using on average only around 2% of the computation time the solver spent to reach the optimal solutions and less than 1% of the computation time the solver spent to prove optimality. Considering that a single charger exists in each CS, we observe that using version N of the subproblem prevents the algorithm from finding solutions to the E-VRP-NL-C for 15 instances. In this case, the algorithm finds the best solution (computed all over our tests) for only 72 of 120 instances. Delaying the starting time of the routes yields better solutions for 20 more instances. In contrast, allowing waiting times in version DW leads to very marginal improvements compared with version D. Indeed, making the EVs wait for an available charger can eliminate capacity violations only if at least

Table 5. Comparison of the Best Results Obtained by the Matheuristic for the E-VRP-NL-C According to the Version of the Subproblem It Uses and the Number of Chargers at Each CS

		Version N			Version D			Version DW			Version DWR		
Capacity	$\|I\|$	#F	#BS	Gap BS	#F	#BS	Gap BS	#F	#BS	Gap BS	#F	#BS	Gap BS
1	10	20	17	0.06%	20	20	0.00%	20	20	0.00%	20	20	0.00%
	20	20	15	0.09%	20	20	0.00%	20	20	0.00%	20	20	0.00%
	40	19	15	0.17%	19	18	0.01%	19	18	0.01%	19	19	0.00%
	80	20	15	0.02%	20	20	0.00%	20	20	0.00%	20	20	0.00%
	160	19	5	0.27%	20	13	0.06%	20	13	0.06%	20	15	0.09%
	320	10	5	0.32%	20	1	0.28%	20	2	0.27%	20	13	0.10%
	All	105	72	0.14%	119	92	0.06%	119	93	0.06%	119	107	0.03%
2	10	20	20	0.00%	20	20	0.00%	20	20	0.00%	20	20	0.00%
	20	20	20	0.00%	20	20	0.00%	20	20	0.00%	20	20	0.00%
	40	19	19	0.09%	20	20	0.00%	20	20	0.00%	20	20	0.00%
	80	20	18	0.01%	20	20	0.00%	20	20	0.00%	20	19	0.01%
	160	20	13	0.05%	20	10	0.18%	20	10	0.18%	20	14	0.09%
	320	19	4	0.33%	20	9	0.12%	20	9	0.12%	20	8	0.17%
	All	118	94	0.08%	120	99	0.05%	120	99	0.05%	120	101	0.05%

Table 6. Comparison of the Average Results Obtained by the Matheuristic for the E-VRP-NL-C According to the Version of the Subproblem It Uses and the Number of Chargers at Each CS

		Version N				Version D				Version DW				Version DWR			
Capacity	$\|I\|$	T	T_1	T_2	Gap BA	T	T_1	T_2	Gap BA	T	T_1	T_2	Gap BA	T	T_1	T_2	Gap BA
1	10	6	5	1	0.06%	6	5	1	0.00%	6	5	1	0.00%	6	5	1	0.00%
	20	11	10	1	0.09%	11	10	1	0.00%	11	10	1	0.00%	11	10	1	0.00%
	40	24	23	1	0.31%	23	22	1	0.03%	23	22	1	0.03%	87	22	65	0.00%
	80	74	73	1	0.10%	74	73	1	0.00%	73	72	1	0.00%	74	73	1	0.04%
	160	340	335	5	0.45%	339	335	4	0.10%	340	336	4	0.10%	397	333	64	0.08%
	320	1345	1274	71	0.34%	1480	1253	227	0.17%	1487	1259	228	0.15%	1933	1244	689	0.05%
	All	300	287	13	0.22%	322	283	39	0.05%	323	284	39	0.05%	418	281	137	0.03%
2	10	5	5	0	0.00%	5	5	0	0.00%	5	5	0	0.00%	5	5	0	0.00%
	20	10	10	0	0.00%	10	10	0	0.00%	10	10	0	0.01%	10	10	0	0.00%
	40	23	23	0	0.05%	23	23	0	0.11%	23	23	1	0.09%	45	23	22	0.01%
	80	73	72	1	0.18%	73	73	1	0.19%	73	73	1	0.25%	73	73	1	0.14%
	160	336	334	2	0.24%	336	334	2	0.25%	337	335	2	0.24%	338	336	2	0.28%
	320	1340	1279	61	0.21%	1283	1276	7	0.12%	1276	1269	7	0.14%	1327	1269	58	0.08%
	All	298	287	11	0.11%	288	287	2	0.11%	287	286	2	0.12%	300	286	14	0.09%

two CSs are visited in a route; otherwise, it is sufficient to make the EVs leave the depot after time 0, as is done in version D. Although the same requirement (more than one CS) holds when allowing the revision of the amount of energy charged at the CSs in version DWR, the latter strategy yields almost all the best solutions to the E-VRP-NL-C.

When increasing the number of chargers to two per CS, the capacity constraints become less binding, and all our assembly strategies yield very similar results. This was somewhat expected but it should be noted that delaying the starting time of the routes increases the probability of ending up with a feasible solution.

The first general conclusion is that allowing delays when solving the CS capacity management subproblem is a suitable and efficient way of assembling high-quality solutions to the E-VRP-NL-C. Allowing the revision of the amounts of energy charged at the CSs on top of it significantly improves the results, but it is slightly more computationally expensive (Table 6). We also note that in general, most of the computation time is spent generating routes as assembling a solution is usually very fast, especially when a cutoff value is provided to the branch-and-cut algorithm. Not surprisingly the computation time increases with the number of customers, but it remains reasonable because it takes around 30 minutes to tackle the largest instances.

To assess the relevance of our cuts, we show in Table 7 the distribution of the different type of cuts generated in the branch-and-cut tree. When the CS capacity constraints are very binding, the energy reasoning-based algorithm detects infeasible route selections much more frequently than the fixed part-based algorithm. We observe that because of its high degree of flexibility, compared with the other versions, DWR requires solving the MILP formulation more often. Indeed, the two previously mentioned algorithms are weaker in that case because the mandatory part of each charging operation may have a very short duration because of the potential revision of the amount of energy charged at the CSs. When we increase the number of chargers at each CS, they lead to the generation of fewer cuts. This is not surprising because they are based on relaxations of the subproblem. However, they are useful because the generated cuts are stronger, which avoids computationally expensive calls to the MILP solver. We also see that integer optimality cuts are seldom generated. Because increases in the total time of the routes are penalized, the algorithm tends to avoid adding waiting times. This indicates that revising the charging operations

Table 7. Characteristics of the Cuts Generating in the Solution Assembly Phase

	Version D			Version DW				Version DWR			
Capacity	FP	NRG	BF	FP	NRG	BF	BO	FP	NRG	BF	BO
1	2.3%	94.5%	3.2%	2.3%	95.0%	2.5%	0.2%	0.6%	16.9%	80.5%	2.0%
2	29.0%	28.2%	42.8%	28.9%	27.1%	42.5%	1.5%	1.3%	0.7%	96.2%	1.8%

Note. FP, feasibility cuts generated after calling the fixed part-based algorithm; NRG, feasibility cuts generated after calling the energy reasoning-based algorithm; BF, feasibility cuts of type generated after solving the MILP formulation; BO, optimality cuts of type generated after solving the MILP formulation.

Table 8. Analysis of the Solutions Retrieved by the Matheuristic

	Version D	Version DW		Version DWR				
Capacity	= D	= D	> W	= D	= NRG	> W	> NRG	> R
1	42.1%	42.2%	1.7%	42.0%	17.4%	1.3%	6.3%	2.5%
2	11.7%	11.7%	0.0%	11.4%	8.5%	0.0%	0.4%	0.0%

Note. D, delay; W, waiting time; NRG, revision of the amount of energy charged; R, removal; =, no increase of the total time; >, increase of the total time.

does not always lead to an increase in time, and a tradeoff between delay and a revision at no cost is often found. To support this conclusion, we analyzed the solutions returned by the matheuristic. Given the number of chargers at each CS, Table 8 reports the percentage of solutions for which there exists at least one route with a delayed starting time, a waiting time before a charging operation, and a revision of the amount of energy charged in the EV. We see that for most of the solutions returned by the algorithm, satisfying the CS capacity constraints can be achieved without increasing the cost.

Because we imposed a CPU time limit for the branch-and-cut execution and because the maximum route duration limit may decrease according to the state of the solution method, no version of the subproblem dominates the other versions. To perform a fair comparison between the different strategies, and to quantify the benefit of allowing the addition of waiting times and the revision of the amount of energy charged at the CSs, we took the 2,400 long-term pools of routes obtained at the end of the matheuristic for version DWR of the subproblem. For every pool of routes, we ran the branch-and-cut algorithm to generate the best E-VRP-NL-C solution. Table 9 reports for each version of the subproblem the number of pool of routes for which the algorithm obtains a feasible solution (#F), as well as the best solution (#Best) among those obtained with the other versions on the same pool. It also shows the average gap between the solution returned and the best solution obtained during these tests under all the assembly strategies. The conclusions are very similar to those drawn from the results of the matheuristic. Compared with version N, version DWR makes nearly 400 extra routes feasible when assuming one charger. Not surprisingly, when increasing the number of chargers, the results tend to be more independent of the selected version of the subproblem.

Last, we compare the BKSs for the E-VRP-NL and those for the E-VRP-NL-C obtained for each instance in our computational experiments. Table 10 reports aggregated results. The second column shows the number of BKSs to the E-VRP-NL that are not feasible for the EVRP-NL-C. As expected, many BKSs to the E-VRP-NL are also BKSs to the E-VRP-NL-C. There is no obvious procedure to identify beforehand whether the CS capacity constraints are binding for a given instance. The other columns focus on the case when the BKSs to the E-VRP-NL-C and the E-VRP-NL are not equal. The third and fourth columns of the table show the number of BKSs that do not have the same number of routes and the average gap in terms of the objective function between the BKSs to the E-VRP-NL-C and the E-VRP-NL. We observe that (i) the introduction of capacity constraints does not increase the number of routes in the vast majority of the cases, and (ii) the increase in the objective function is limited. The fifth and sixth columns of the table show the percentage of routes that make up the BKSs to both problems with and without capacity (differentiating the case when the starting time is only delayed). Although the CS capacity is not problematic in many instances, when it is, the solutions differ widely.

When a violation of the CS capacity constraints is observed for a BKS to the E-VRP-NL, we tested the use of version DWR of the capacity management subproblem to resolve it. We observed that 31 of the 53 solutions and 8 of the 12 solutions can be repaired into a feasible solution to the E-VRP-NL-C when the capacity is equal to one and two, respectively. We conclude that a risk is incurred if CS capacity constraints are only considered a posteriori.

Table 9. Comparison of the Results of the Branch-and-Cut Algorithm According to the Different Versions of the Subproblem

	Version N			Version D			Version DW			Version DWR		
Capacity	#F	#Best	Gap	#F	#Best	Gap	#F	#Best	Gap	#F	#Best	Gap
1	1916	1361	0.315%	2262	2095	0.096%	2270	2112	0.091%	2307	2307	0.000%
2	2321	2125	0.073%	2390	2371	0.013%	2390	2371	0.013%	2390	2390	0.000%

Table 10. Comparison Between the Best-Known Solutions to the E-VRP-NL and E-VRP-NL-C

Capacity	Number of instances where the BKS to the E-VRP-NL is not feasible for the E-VRP-NL-C	Number of instances where the two BKS have a different number of routes	Increase in the objective function value	Percentage of routes		
				=	Delayed	≠
1	53/119[a]	4/53	0.29%	36%	10%	54%
2	12/120	1/12	0.16%	38%	7%	55%

[a]No solution is known for a particular instance.

6. Conclusion and Perspectives

We modeled and solved an EV routing problem that embeds several features such as piecewise linear charging functions, multiple charging technologies, and multiple stops at CSs. Our methodology explicitly considers the fact that the number of EVs simultaneously charging at every CS is limited by the number of chargers. We have proposed a continuous-time path-based formulation of the E-VRP-NL-C. Our results show that optimally solving even small-size instances of the problem with a commercial MILP solver running this formulation is challenging. To handle larger instances, we have developed an algorithmic framework which iteratively calls a route generator and a solution assembler. The first component focuses on generating a pool of high-quality and diversified routes from solutions to the E-VRP-NL obtained by means of an ILS metaheuristic. Computational experiments have shown that this first component produces high quality E-VRP-NL solutions. Indeed, we improved 80 of 120 best-known E-VRP-NL solutions. The second component assembles a solution to the E-VRP-NL-C by selecting a subset of routes from the generated pool. We decomposed this assembly problem into a route selection problem and a CS capacity management subproblem. We designed a branch-and-cut algorithm based on this decomposition scheme. We developed and compared four versions of the CS capacity management subproblem, including a simple check of the capacity constraints, the introduction of waiting times, and the revision of the charging amounts. Our results show that there exists a serious risk of ending up with no solution if the method disregards the CS capacity constraints or if the routes cannot be modified by the subproblem. Delaying the starting time and revising the amount of energy charged at the visited CSs lessens this risk. Using more complex strategies to solve capacity violation issues at CSs does not significantly increase the computation time and yields better solutions. The matheuristic is also able to find all optimal solutions of small-size instances.

Although our work focuses on CSs that are privately operated, it also applies if reserving chargers at CSs is possible (which is very rare in practice). In this case, there may be time windows when chargers cannot be reserved. We can deal with these periods of unavailability in our matheuristic by inserting dummy operations (fixed in time) in the capacity management subproblem and by slightly adapting the labeling algorithm used to insert charging decisions in the ILS.

References

Alvarenga GB, Mateus G, De Tomi G (2007) A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Comput. Oper. Res.* 34(6):1561–1584.

Andelmin J, Bartolini E (2017) An exact algorithm for the green vehicle routing problem. *Transportation Sci.* 51(4):1288–1303.

Andelmin J, Bartolini E (2019) A multi-start local search heuristic for the Green Vehicle Routing Problem based on a multigraph reformulation. *Comput. Oper. Res.* 109:43–63.

Artigues C, Michelon P, Reusser S (2003) Insertion techniques for static and dynamic resource-constrained project scheduling. *Eur. J. Oper. Res.* 149(2):249–267.

Baptiste P, Le Pape C, Nuijten W (2001) *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*, vol. 39 (Springer, Boston, MA).

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.

Brandstätter G, Leitner M, Ljubić I (2020) Location of charging stations in electric car sharing systems. *Transportation Sci.* 54(5): 1408–1438.

Bruglieri M, Mancini S, Pisacane O (2019a) The green vehicle routing problem with capacitated alternative fuel stations. *Comput. Oper. Res.* 112:104759.

Bruglieri M, Mancini S, Pisacane O (2019b) More efficient formulations and valid inequalities for the Green Vehicle Routing Problem. *Transportation Res., Part C Emerging Tech.* 105:283–296.

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

Bruglieri M, Mancini S, Pezzella F, Pisacane O (2019) A path-based solution approach for the Green Vehicle Routing Problem. *Comput. Oper. Res.* 103:109–122.

Codato G, Fischetti M (2006) Combinatorial Benders' cuts for mixed-integer linear programming. *Oper. Res.* 54(4):756–766.

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

Drexl M (2012) Synchronization in vehicle routing: A survey of vrps with multiple synchronization constraints. *Transportation Sci.* 46(3):297–316.

El Hachemi N, Gendreau M, Rousseau LM (2013) A heuristic to solve the synchronized log-truck scheduling problem. *Comput. Oper. Res.* 40(3):666–673.

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

Felipe A, 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.

Gnann T, Funke S, Jakobsson N, Plötz P, Sprei F, Bennehag A (2018) Fast charging infrastructure for electric vehicles: Today's situation and future needs. *Transportation Res. Part D Transportation Environment* 62:314–329.

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

Grangier P, Gendreau M, Lehuédé F, Rousseau LM (2017) A matheuristic based on large neighborhood search for the vehicle routing problem with cross-docking. *Comput. Oper. Res.* 84:116–126.

Grimault A, Bostel N, Lehuédé F (2017) An adaptive large neighborhood search for the full truckload pickup and delivery problem with resource synchronization. *Comput. Oper. Res.* 88:1–14.

Hempsch C, Irnich S (2008) Vehicle routing problems with inter-tour resource constraints. Golden B, Raghavan S, Wasil E, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges* (Springer, Boston, MA), 421–444.

Hiermann G, Hartl RF, Puchinger J, Vidal T (2019) Routing a mix of conventional, plug-in hybrid, and electric vehicles. *Eur. J. Oper. Res.* 272(1):235–248.

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

Juan AA, Mendez CA, Faulin J, de Armas J, Grasman SE (2016) Electric vehicles in logistics and transportation: A survey on emerging environmental, strategic, and operational challenges. *Energies* 9(2):86.

Keskin M, Çatay B (2016) Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Res., Part C Emerging Tech.* 65:111–127.

Keskin M, Çatay B (2018) A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Comput. Oper. Res.* 100:172–188.

Keskin M, Laporte G, Çatay B (2019) Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Comput. Oper. Res.* 107:77–94.

Koç Ç, Karaoglan I (2016) The green vehicle routing problem: A heuristic based exact solution approach. *Appl. Soft Comput.* 39: 154–164.

Koç Ç, Jabali O, Mendoza JE, Laporte G (2018) The electric vehicle routing problem with shared charging stations. *Internat. Transportation Oper. Res.* 26(4):1211–1243.

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

Lam E, Van Hentenryck P (2016) A branch-and-price-and-check model for the vehicle routing problem with location congestion. *Constraints* 21(3):1–19.

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

Leggieri V, Haouari M (2017) A practical solution approach for the green vehicle routing problem. *Transportation Res., Part E Logist. Transportation Rev.* 104:97–112.

Lourenço HR, Martin OC, Stützle T (2003) Iterated local search. Glover F, Kochenberger GA, eds. *Handbook of Metaheuristics*, vol. 57 (Springer, Boston, MA), 320–353.

Macrina G, Di Puglia Pugliese L, Guerriero F, Laporte G (2019) The green mixed fleet vehicle routing problem with partial battery recharging and time windows. *Comput. Oper. Res.* 101:183–199.

Montoya A, Guéret C, Mendoza J, Villegas J (2016) A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Res., Part C Emerging Tech.* 70:113–128.

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.

Morais VWC, Mateus GR, Noronha TF (2014) Iterated local search heuristics for the vehicle routing problem with cross-docking. *Expert Systems Appl.* 41(16):7495–7506.

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

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

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.

Rix G, Rousseau LM, Pesant G (2015) A column generation algorithm for tactical timber transportation planning. *J. Oper. Res. Soc.* 66(2):278–287.

Roberti R, Wen M (2016) The electric traveling salesman problem with time windows. *Transportation Res., Part E Logist. Transportation Rev.* 89:32–52.

Sassi O, Oulamara A (2014) Joint scheduling and optimal charging of electric vehicles problem. Murgante B, Misra S, Rocha AMAC, Torre C, Rocha JG, Falcão MI, Taniar D, et al, eds. *Computational Science and Its Applications*, vol. 8580 (Springer, Cham, Switzlerland), 76–91.

Schiffer M, Schneider M, Laporte G (2018) Designing sustainable mid-haul logistics networks with intra-route multi-resource facilities. *Eur. J. Oper. Res.* 265(2):517–532.

Schiffer M, Stütz S, Walther G (2018) Electric commercial vehicles in mid-haul logistics networks. Pistoia G, Liaw B, eds. *Behaviour of Lithium-Ion Batteries in Electric Vehicles* (Springer, Cham, Switzerland), 153–173.

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

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

Subramanian A, Uchoa E, Ochi LS (2013) A hybrid algorithm for a class of vehicle routing problems. *Comput. Oper. Res.* 40(10):2519–2531.

Villegas JG, Guéret C, Mendoza JE, Montoya A (2018) The technician routing and scheduling problem with conventional and electric vehicle, hal-01813887. Accessed 12/08/2021. https://hal.archives-ouvertes.fr/hal-01813887.

Villegas JG, Prins C, Prodhon C, Medaglia AL, Velasco N (2013) A matheuristic for the truck and trailer routing problem. *Eur. J. Oper. Res.* 230(2):231–244.

Zachariadis EE, Kiranoudis CT (2010) A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Comput. Oper. Res.* 37(12): 2089–2105.