Mobility-on-Demand with Electric Vehicles: Scalable Route and Recharging Planning through Column Generation

Axel Parmentier\textsuperscript{1*}, Rafael Martinelli\textsuperscript{2}, Thibaut Vidal\textsuperscript{3,4}

\textsuperscript{1} Cermics, École des Ponts, Marne la Vallée, France
\textsuperscript{2} Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro
\textsuperscript{3} Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
\textsuperscript{4} Département de mathématiques et de génie industriel, Polytechnique Montréal

Abstract. The rise of battery-powered vehicles for mobility-on-demand leads to many technical and methodological hurdles. Among these, the efficient planning of a shared electric fleet to fulfill passenger transportation requests still represents a major challenge. This is due to the specific constraints of electric vehicles, bound by their autonomy and necessity of recharge planning, and the typical large scale of these systems, which challenges existing optimization algorithms. The purpose of this paper is to introduce a scalable column generation approach for the problem of managing electric mobility-on-demand fleets. Our algorithm relies on three ingredients: (i) an algebraic framework leading to an efficient algorithm for the pricing subproblem despite its non-linearity (ii) sparsification approaches permitting to decrease the size of the subjacent graphs dramatically, and (iii) a diving heuristic, which locates near-optimal solutions in a fraction of the time needed for a complete branch-and-price. Through extensive computational experiments, we demonstrate that our approach significantly outperforms previous algorithms for this setting, leading to accurate solutions for problems counting several hundreds of requests.

Keywords. Mobility-on-demand, Routing and scheduling, Electric vehicles, Column generation, Diving heuristics.

* Corresponding author: axel.parmentier@enpc.fr

1 Introduction

To cut down pollution and carbon footprint, major cities worldwide have established regulations to progressively phase out conventional internal combustion engine (ICE) vehicles and replace them with subsidized electric vehicles (EV). Public bus transportation has already shifted towards battery-powered vehicles in many cities (Li et al. 2016, Mahmoud et al. 2016, Wang et al. 2017). In contrast, taxi companies and mobility-on-demand services take more time to transition as they often depend on the driver’s willingness to switch to EVs. Still, EV fleets are becoming more widespread in various cities (see, e.g., Dunne 2017, De Jong 2018, London Authorities 2018, Hu et al. 2018, Scorrano et al. 2020). This ongoing transformation has led
to new market segments focused on emission-free trips, and major ride-hailing companies roll out specific plans to provide advantageous leasing conditions, dedicated priority queues, and charging infrastructures at airports for EV drivers.

Despite all the actions taken, the electric mobility-on-demand (EMoD) sector’s growth is subject to different constraints than other sectors. Whereas buses operate on a fixed itinerary and require fast-charging (or battery-swap) stations in a limited number of locations, EMoD services require scattered charging stations to operate efficiently. Intense competition also occurs between taxi and ride-hailing companies, such that efficient planning of customer-to-vehicle assignments, dead-heading trips, and recharging actions is critical for customer satisfaction and profitability. When the charging infrastructure is established, the optimization of vehicle itineraries and recharging trips to meet the demand for a set of timetabled trips can be formalized as an electric vehicle scheduling problem (EVSP – Bodin et al. 1983, Wen et al. 2016). It is common to register hundreds of transportation requests in densely populated areas, leading to planning problems of large size that call for new solution paradigms.

This paper contributes to overcoming the existing methodological gaps and the need for efficient and scalable solution algorithms. It introduces exact and heuristic column generation algorithms for the EVSP grounded on the monoid pricing paradigm by Parmentier (2019). The proposed pricing algorithm is a labeling algorithm that exploits an algebraic constraint model to build high-quality bounds on path resources and ultimately to reduce the number of paths. To further enhance efficiency on large-scale cases, we apply sparsification techniques to the original graph and restrict it to a much smaller subset of promising arcs. Then, we embed the column generation in a specialized diving algorithm that uses strong diving up to a given depth. As seen in our experiments, these techniques permit us to efficiently solve large-scale problems with several hundreds of requests. The contributions of this paper therefore are:

1. A new monoid pricing algorithm for EVSPs, leading to the first mathematical programming algorithm able to solve all existing 100-customers instances for the problem.

2. Tailored sparsification and diving strategies, which permits scaling up the algorithm to cases counting half a thousand visits and finding near-optimal solutions in those cases.

3. An extensive numerical campaign, which evaluates the impact of critical methodological components and demonstrates the method’s applicability on a wide range of instances.

4. More generally, this work paves the way towards a new generation of scalable algorithms for EVSPs.

The remainder of this paper is as follows. Section 2 formally defines the EVSP and reviews the existing literature. The proposed monoid pricing algorithm is described in Section 3, and Section 4 explains how this algorithm can be integrated into an exact branch-and-price framework and a heuristic column generation algorithm. Section 5 describes our computational experiments, and Section 6 finally concludes.
2 Problem Statement and Literature Review

Problem Statement. Based on the same problem conventions as in Wen et al. (2016), the EVSP can be defined as follows. Let $\mathcal{T}$ be a set of timetabled passenger trips that a company should operate on a given day, let $\mathcal{D}$ be the set of depots where the vehicles start and end every day, and let $\mathcal{S}$ be a set of stations where vehicles can charge their battery. Each trip $\tau \in \mathcal{T}$ is planned to start at $t^b_\tau \in \mathbb{R}$, end at $t^r_\tau \in \mathbb{R}$, and consumes a quantity $e_\tau$ of energy. We define the set of locations $\mathcal{Z} = \mathcal{T} \cup \mathcal{D} \cup \mathcal{S}$ and set $e_z = 0$ for $z \in \mathcal{D} \cup \mathcal{S}$ for notation convenience. For each location pair $z_1, z_2$ in $\mathcal{Z}$, we denote respectively by $\delta_{z_1,z_2}$, $c_{z_1,z_2}$, and $e_{z_1,z_2}$ the driving time, cost, and energy consumption required to go from the location of (the end of) $z_1$ to the location of (the beginning of) $z_2$. Times belong to $[0, t^{\text{max}}]$, where 0 and $t^{\text{max}}$ respectively correspond to the beginning and the end of the day.

The number of vehicles stationed at each depot is not limited, but each vehicle is characterized by a fixed use cost $c_{\text{veh}}$ and equipped with a battery of capacity $M^{\text{ch}}$. Batteries can be recharged at any charging station, in such a way that it takes a time $e/\alpha$ to increase the charge of the battery by $e$.

A vehicle route $r$ is a sequence $z_0, \ldots, z_{k^r}$ of elements of $\mathcal{Z}$ such that $z_0 \in \mathcal{D} \cup \mathcal{T}$, $z_i \in \mathcal{S} \cup \mathcal{T}$ for $1 \leq i \leq k^r - 1$, $z_{k^r} \in \mathcal{D} \cup \mathcal{T}$, and there exists $t^b_{z_i}$ and $t^r_{z_i}$ in $[0, t^{\text{max}}]$ for each $i$ such that

$$t^b_{z_i} = t^r_{z_{i-1}} + \delta_{z_{i-1},z_i} \quad \text{for } i > 1 \quad \text{and} \quad \begin{cases} \quad t^b_{z_i} = t^b_{z_i} \quad \text{and} \quad t^r_{z_i} = t^b_{z_i} \quad \text{if } z_i \in \mathcal{T} \\ \quad t^b_{z_i} \leq t^r_{z_i} \quad \text{otherwise.} \end{cases} \quad (1)$$

If $z_i$ is a station, a vehicle operating $r$ arrives in the station at $t^b_{z_i}$ and leaves it at $t^r_{z_i}$. It leaves the depot at $t^b_{z_0}$ and returns to it at $t^b_{z_{k^r}}$. Given an initial level $\ell^\text{in}$ in $[0, M^{\text{ch}}]$, a route scheduling $t' = (t^b_{z_i}, t^r_{z_i})_{i \in [k^r]}$, we recursively define the battery level $\ell^r_{z_i}(\ell^\text{in}, t')$ at the beginning of activity $z_i$ given $\ell^\text{in}$ as

$$\ell^r_{z_i}(\ell^\text{in}, t') = \begin{cases} \ell^\text{in} & \text{if } i = 0, \\ \ell^r_{z_{i-1}}(\ell^\text{in}, t') - e_{z_{i-1},z_i} & \text{if } i > 0 \text{ and } z_{i-1} \in \mathcal{T}, \\ \min(M^{\text{ch}}, \ell^r_{z_{i-1}}(\ell^\text{in}, t') + \alpha(t^r_{z_{i-1}} - t^b_{z_{i-1}})) & \text{if } i > 0 \text{ and } z_{i-1} \in \mathcal{S}. \end{cases} \quad (2)$$

A route is feasible given $\ell^\text{in}$ if there exists a scheduling $t'$ of $r$ such that $\ell^r_{z_i}(\ell^\text{in}, t') \geq e_{z_i}$ for all $i$ in $[k^r]$. It is feasible if it is feasible for $M^{\text{ch}}$. Equation (1) ensures that $t^b_{z_i}$ and $t^r_{z_i}$ are non-decreasing functions of $i$, which implies that a given trip $\tau \in \mathcal{T}$ appears at most once in a route $r$. A complete route is a feasible route $z_0, \ldots, z_{k^r}$ such that $z_0 = z_{k^r} \in \mathcal{D}$. The cost of this route is defined as

$$c_r = c_{\text{veh}} + \sum_{i=1}^{k^r} c_{z_{i-1},z_i},$$

where the first term represents the fixed use cost of the vehicle, and the second term stands for the driving cost. Remark that $c_r$ does not depend on the choice of $t^b_{z_i}$ in Equation (1). We denote by $\tau \in r$ the fact that the timetabled trip $\tau$ belongs to route $r$. We denote by $\mathcal{R}$ and $\mathcal{R}^c$
the set of feasible routes and the set of complete routes. Moreover, observe that consecutive visits to charging stations are allowed.

With these definitions, the goal of the EVSP is to find a set \( R \) of feasible complete routes in \( \mathcal{R}^c \) covering each trip and such that the sum of the costs of the routes in \( R \) is minimum. Given \( x_r \), a binary variable indicating if a route \( r \) is in the solution, the electric vehicle scheduling problem can be stated as the following master problem:

\[
\begin{align*}
\min \quad & \sum_{r \in \mathcal{R}} c_r x_r \\
\text{s.t.} \quad & \sum_{r \ni \tau} x_r = 1 \quad \forall \tau \in \mathcal{T} \\
\quad & x_r \in \{0, 1\} \quad \forall r \in \mathcal{R}^c.
\end{align*}
\]

Naturally, the size of \( \mathcal{R}^c \) grows exponentially with the number of timetabled trips, making any direct enumeration impractical in most practical cases. We will rely on column generation (Barnhart et al. 1998) to circumvent this issue.

Related Literature. Early studies on vehicle scheduling problems (VSP) date back to Bodin et al. (1983) and often arose from the air transportation literature. The term “vehicle scheduling” has been coined to describe routing problems in the presence of determined start dates for the trips, therefore establishing a total order. Solution algorithms for VSPs are typically more scalable than their vehicle routing problem (VRP) counterparts since they are not dependent on cycle elimination. A similar situation happens when trip durations are greater than the width of their time windows. Consequently, the canonical VSP with a single depot and no other constraint than the customers’ service times is known to be polynomially-solvable as a minimum-cost flow problem in an acyclic digraph. However, many immediate extensions of the problem, e.g., with multiple depots or duration constraints on the trip, belong to the NP-hard class. Among all variants surveyed in Bodin et al. (1983) and Bunte and Kliewer (2009), the VSP with “length of path considerations” (i.e., duration limits) is particularly relevant to our problem. Duration constraints model range-limitations which are typical in electric vehicles, but do not allow for possible recharging. This variant has been solved by Haghani and Banihashemi (2002), Ribeiro and Soumis (1994), and Desrosiers et al. (1995) with heuristics and column-generation algorithms.

More recently, VSPs with duration constraints and possible recharging stops have regained attention due to their economic significance in electric mobility systems. Adler and Mirchandani (2017) considered the VSP with alternative-fuel vehicles and multiple depots, in which a limited number of nodes act as charging stations. This model only allows full charging and supposes that the charging time is constant, regardless of the energy level upon arrival to the station. A similar model is studied in Li (2014). Constant-time charging is relevant when considering battery-swapping technologies, but only represents a coarse approximation of mobility systems based on charging technology. Therefore, Wen et al. (2016) define the electric VRP (EVSP),
which integrates the possibility of partial charging in a time which grows linearly with the amount of electricity recharged. Finally, van Kooten Niekerk et al. (2017) propose a more accurate and general charging model, considering non-linear charging rates as well as possible battery-deterioration effects. Due to the inherent complexity of this charging scheme, discretization techniques are used for the more complex models.

A similar progression is observable when surveying the literature about electric vehicle routing problems (EVRP). As illustrated in the surveys of Brandstätter et al. (2016), Pelletier et al. (2016), Schiffer et al. (2019) and Vidal et al. (2020), after a first wave of studies focused on full constant-time charging, research have progressed towards more sophisticated and realistic problem settings, e.g., with mixed vehicle fleets (Hiermann et al. 2016, 2019), richer delivery networks (Schiffer and Walther 2017, Breunig et al. 2019), or considering heterogeneous charging infrastructures (Felipe et al. 2014, Keskin and Çatay 2018) and non-linear charging rates (Montoya et al. 2017, Pelletier et al. 2017). However, the resulting models are highly complex to solve, such that the vast majority of these articles propose metaheuristic approaches. Moreover, solution methods are tested on benchmark instances that rarely exceed a hundred customers.

Specific to exact methods, with the exception Wen et al. (2016), which is based on a mixed integer programming (MIP) formulation and branch-and-cut algorithm, all state-of-art algorithms for EVSP variants (Li 2014, Adler and Mirchandani 2017, van Kooten Niekerk et al. 2017) rely on a Dantzig-Wolfe decomposition of the set partitioning formulation, and therefore repeatedly solve a pricing problem which can be formulated as a variant of resource-constrained shortest path problem (RCSP) (Irnich and Desaulniers 2005). When only full constant-time charging is allowed, the pricing problem is a particular case of the weight-constrained shortest path with replenishment arcs, studied in Smith et al. (2012) and Bolívar et al. (2014). More realistic charging assumptions (e.g., partial charging) lead to more complex RCSPs. In most state-of-the-art EVSP and EVRP column generation algorithms (Desaulniers et al. 2016), pricing problems are solved with labeling techniques, through resource extension functions. Bi-directional search algorithms have also been recently developed (Tilk et al. 2017). However, the aforementioned algorithms still do not consistently solve instances with more than a hundred customers for this problem class. To fill this gap, we stray away from the usual approach based on resource extension functions and explore instead a monoid pricing approach grounded on the work of Parmentier (2019). Our approach involves defining an algebraic constraint model to find tight resource bounds. Moreover, to scale up the solution approach to large-scale problems arising in EMoD settings, we design efficient sparsification techniques and diving heuristics, allowing a fast heuristic column generation with only a minor loss of solution quality. Our solution approach will be described in the next sections, starting with the solution of the pricing problem and following with the solution algorithms.

3 Pricing Subproblem

As we consider only complete routes that start and end in the same depot, the pricing subproblem can be solved separately for each depot $\theta \in \mathcal{D}$. Therefore, we assume that a depot $\theta$ is selected
and explain how to solve the pricing subproblem:

$$\min_{r \in \mathcal{R}_\theta} c_r - \sum_{\tau \in \mathcal{R}_\theta} \lambda_\tau,$$

where $\mathcal{R}_\theta$ is the set of routes of $\mathcal{R}_\theta$ that start and end in $\theta$. To that purpose, we reduce our problem to a **Monoid Resource Constrained Shortest Path Problem** and exploit the algorithms recently introduced in Parmentier (2019) for that problem. For presentation convenience, we use the same notations as in Parmentier and Meunier (2020).

### 3.1 Framework and Algorithm

A binary operation $\oplus$ on a set $M$ is **associative** if $q \oplus (q' \oplus q'') = (q \oplus q') \oplus q''$ for $q, q', q''$ in $M$. An element 0 is **neutral** if $0 \oplus q = q \oplus 0 = q$ for any $q$ in $M$. A set $(M, \oplus)$ is a **monoid** if $\oplus$ is associative and admits a neutral element. A partial order $\preceq$ is **compatible** with $\oplus$ if the mappings $q \mapsto q \oplus q'$ and $q \mapsto q' \oplus q$ are non-decreasing according to this order for all $q'$ in $M$. A partially ordered set $(M, \preceq)$ is a **lattice** if any pair $(q, q')$ of elements of $M$ admits a greatest lower bound or **meet** denoted by $q \wedge q'$, and a least upper bound or **join** denoted by $q \vee q'$. A set $(M, \oplus, \preceq)$ is a **lattice-ordered monoid** if $(M, \oplus)$ is a monoid, $(M, \preceq)$ is a lattice and $\preceq$ is compatible with $\oplus$.

Given a digraph $D = (V, A)$, a lattice-ordered monoid $(M, \oplus, \preceq)$, elements $q_a \in M$ for each $a \in A$, origin and destination vertices $o$ and $d$, and two non-decreasing mappings $c : M \to \mathbb{R}$ and $\rho : M \to \{0, 1\}$, the **Monoid Resource Constrained Shortest Path Problem** seeks an $o$-$d$ path $P$ of minimum $c\left(\bigoplus_{a \in P} q_a\right)$ among those satisfying $\rho\left(\bigoplus_{a \in P} q_a\right) = 0$,

where $\bigoplus_{a \in P}$ is always performed in the order of the arcs on the path $P$ (the operation $\oplus$ is not necessarily commutative). We call such a $q_a$ the **resource** of the arc $a$. The sum $\bigoplus_{a \in P} q_a$ is the **resource** of a path $P$, and we denote it by $q_P$. The real number $c(q_P)$ is its **cost**, and the path $P$ is **feasible** if $\rho(q_P)$ is equal to 0. Therefore we call $c$ and $\rho$ the **cost** and the **infeasibility functions**.

We now describe an **enumeration algorithm** for the **Monoid Resource Constrained Shortest Path Problem**. It follows the standard labeling scheme (Irnich and Desaulniers 2005) for resource-constrained shortest paths. The specificity of our algorithm is that it uses, for each $v$ in $V$, a lower bound

$$b_v \preceq q_P \quad \text{for any } v\text{-}d \text{ path } P.$$  

Having defined these bounds, we define $\text{key}(P)$ as

$$\text{key}(P) = c(q_P \oplus b_v) \quad \text{where } v \text{ is the last vertex of } P.$$  

6
The empty path at a vertex $v$ is the path with no arcs starting and ending at $v$. By definition of path resources, its resource is the neutral element of the monoid. A path $P$ dominates a path $Q$ if $q_P \preceq q_Q$. During the algorithm, a list $L$ of partial paths, an upper bound $U^B_{od}$ on the cost of an optimal solution, and lists $(L^d_v)_{v \in V}$ of non-dominated $o$-$v$ paths are maintained. Algorithm 1 states our algorithm. We denote by $P + a$ the path composed of a path $P$ followed by an arc $a$.

**Algorithm 1** Enumeration algorithm for the Monoid Resource Constrained Shortest Path Problem

1. **input**: bounds $(b_v)_{v \in V}$ satisfying (5); (See Section 3.1.1)
2. **initialization**: $U^B_{od} \leftarrow +\infty$, $L \leftarrow \emptyset$, and $L^d_{v} \leftarrow \emptyset$ for each $v \in V$;
3. add the empty path at the origin $o$ to $L$ and $L^d_o$;
4. **while** $L$ is not empty **do**
5. $P \leftarrow$ a path of minimum key($P$) in $L$;
6. $L \leftarrow L \setminus \{P\}$;
7. $v \leftarrow$ last vertex of $P$;
8. **if** $v = d$, $\rho(q_P) = 0$, and $c(q_P) < U^B_{od}$ **then**
9. $U^B_{od} \leftarrow c(q_P)$;
10. **else**
11. **for all** $a \in \delta^+(v)$ **do**
12. $Q \leftarrow P + a$;
13. $w \leftarrow$ last vertex of $Q$;
14. **if** $\rho(q_Q \oplus b_w) = 0$ and $c(q_Q \oplus b_w) < U^B_{od}$ **then**
15. **if** $Q$ is not dominated by any path in $L^d_{w}$ **then**
16. $L^d_{w} \leftarrow L^d_{w} \cup \{Q\}$ and remove from $L^d_{w}$ and $L$ every path dominated by $Q$;
17. $L \leftarrow L \cup \{Q\}$;
18. **end if**
19. **end if**
20. **end for**
21. **end if**
22. **end while**
23. **return** $U^B_{od}$;

**Proposition 1.** Suppose that $D$ is acyclic. Then Algorithm 1 converges after a finite number of iterations, and, at the end of the algorithm, $U^B_{od}$ is equal to the cost of an optimal solution of the Monoid Resource Constrained Shortest Path Problem if such a solution exists, and to $+\infty$ otherwise.

Our algorithm is a variant of the label correcting algorithm for resource-constrained shortest path problem, using the bounds of Equation (5) in the algorithm. While it is well known that the use of lower bounds is a critical element in the performance of the enumeration algorithms (Dumitrescu and Boland 2003), our approach enables us to use such bounds to model non-linear constraints such as electricity consumption with partial recharge. Not only are these bounds used to discard more paths, but they are also used to improve the order in which the algorithm considers the paths.
3.1.1 Bounds Computation

As input of Algorithm 1, we use the bounds $b_v$ are defined as follows

$$b_v = \begin{cases} 
0 & \text{if } v = d, \\
\bigwedge_{a=(v,w)\in\delta^+(v)} q_a \oplus b_w & \text{otherwise.} 
\end{cases} \quad (7)$$

When $(M, \oplus, \preceq)$ is equal to $\mathbb{R}$, then $\bigwedge$ is the minimum and $\oplus$ the usual sum on $\mathbb{R}$, Equation (7) is the usual dynamic programming equation on acyclic digraphs, and $b_v$ is the cost of a shortest $v$-$d$ path. We can compute the $b_v$ for a generic $(M, \oplus, \preceq)$ with the usual dynamic programming algorithm on a reverse topological order. When $\preceq$ is a general partial order, then $\bigwedge$ is no more a minimum, as it can “mingle” resources from several outgoing arcs. Still, an induction along a reverse topological order enables us to prove that the $b_v$ obtained satisfy (5) (Parmentier 2019).

3.2 Energy Consumption along Routes

We now introduce some results on battery consumption along routes that will be helpful to reduce the pricing subproblem to a MONOID RESOURCE CONSTRAINED SHORTEST PATH PROBLEM in the next section.

Before starting, we underline that there can be several stations in a row in-between depots and trips. A route $r = z_0, \ldots, z_{kr}$ is a station sequence between $z_0$ and $z_{kr}$ if $z_0$ and $z_{kr}$ are in $D \cup T$, and $z_i$ is a station in $S$ for $i$ in $\{1, \ldots, k^r - 1\}$.

3.2.1 Optimal Station Sequence and Route Scheduling

Given a feasible route $r = z_0, \ldots, z_{kr}$, Equation (1) lets some flexibility in the choice of $t_{z_i}^{b,r}$ and $t_{z_i}^{e,r}$ if $z_i$ is a station. Moreover, the choice of the route schedule $t^r$ impacts the charge $\ell_{z_{kr}}^r(\ell^{in})$ at the end of the route. We now characterize a schedule $t^r$ that enables to maximize $\ell_{z_{kr}}^r(\ell^{in})$.

Given a route $r = z_0, \ldots, z_{kr}$ and an initial charge $\ell^{in}$, we define $t^{r,*}(\ell^{in})$ as follows. Departure and arrival time in stations are fixed. If $z_0$ is a depot, we define $t_{z_0}^{b,r,*} = t_{z_0}^{e,r,*} = 0$. If $z_{kr}$ is a depot, we define $t_{z_{kr}}^{b,r,*} = t_{z_{kr}}^{e,r,*} = t_{\text{max}}$. For $i$ in $\{1, \ldots, k - 1\}$, $z_i$ is a station and we define recursively $t_{z_i}^{b,r,*}(\ell^{in}) = t_{z_i}^{e,r,*}(\ell^{in}) + \delta_{z_{i-1},z_i}$ and

$$t_{z_i}^{e,r,*}(\ell^{in}) = \min \left( t_{z_i}^{b,r,*}(\ell^{in}) + \frac{M_{\text{ch}} - \ell_{z_i}^r(\ell^{in}, t^{r,*}(\ell^{in}))}{\alpha}, t_{z_{kr}}^{b} - \sum_{j=i+1}^{k^r} \delta_{z_{j-1},z_j} \right).$$

The following lemma shows that route schedule $t^{r,*}(\ell^{in})$ is optimal.

**Proposition 2.** For any route $r = z_0, \ldots, z_{kr}$ and initial charge $\ell^{in}$, we have

$$\ell_{z_{kr}}(\ell^{in}, t^{r,*}(\ell^{in})) \geq \ell_{z_{kr}}(\ell^{in}, t^r) \quad \text{for any } t^r \text{ satisfying (1)}.$$
Proof. We start by proving the result for station sequences $r$. The time that can be spent charging during the station sequence $r$ is $t_{z_k}^b - t_{z_0}^e - \sum_{i=1}^{k} \delta_{z_{i-1},z_i}$, the remaining time being spent driving. Hence, denoting $e_{nc}^r = \alpha \left(t_{z_k}^b - t_{z_0}^e - \sum_{i=1}^{k} \delta_{z_{i-1},z_i}\right)$, an upper bound on the battery level that can be reached at the end of the last station of the sequence is

$$\begin{cases}\
\min \left(M_{ch}, \ell_{in} + e_{nc}^r - \sum_{i=1}^{k-1} e_{z_{i-1},z_i}\right), & \text{if } z_0 \text{ in } D, \\
\min \left(M_{ch}, \ell_{in} + e_{nc}^r - e_{z_0} - \sum_{i=1}^{k-1} e_{z_{i-1},z_i}\right), & \text{if } z_0 \text{ in } S.
\end{cases}$$

(8)

Scheduling $t^r(\ell_{in})$ enables to achieve this bound.

We now extend the result from station sequences to general routes route $r$. As a route can be decomposed into station sequences separated by trips $\tau$ with fixed starting and ending times, it suffices to use $t^r(\ell_{in})$ recursively on station sequences, level $\ell_{in}$ at the beginning of a station sequence being equal to the level at the end of the previous station sequence.

From now on, we assume that routes are scheduled optimally and define

$$\ell^r_z(\ell_{in}) = \ell^r_z(\ell_{in}, t^r).$$

3.2.2 Minimum Charge at the Beginning, Maximum Charge at the End

We introduce quantities that will later be used in the reduction to the Monoid Resource Constrained Shortest Path Problem in the next section. Given a feasible route $r$, we define $\ell^{r}_{mb}$ as the minimum initial charge $\ell_{in}$ such that $r$ is feasible given $\ell_{in}$, $\ell^r_{Me}$ as the charge $\ell^{r}_{z_k}(M_{ch})$ at the end of $r$ if the battery is fully charged at the beginning of $r$, and $\Delta \ell^r$ as the difference $\ell^{r}_{z_k}(\ell^r_{mb}) - \ell^r_{mb}$ between the charge at the end $r$ and the charge at the beginning if the vehicle is initially at the minimum charge $\ell^r_{mb}$. Those quantities are illustrated in Figure 1.

![Figure 1: Battery level as a function of time on route $r = \tau_1, \tau_2, \sigma, \tau_3$.](image-url)
**Proposition 3.** Let \( r_1 \) and \( r_2 \) be feasible routes respectively ending and starting in \( \tau \), and let \( r \) be the route composed of \( r_1 \) followed by \( r_2 \). Then \( r \) is feasible if and only if \( \ell_{r_1}^r \geq \ell_{r_2}^r \). And if \( r \) is feasible, we have

\[
\begin{align*}
\ell_{mb}^r &= \max(\ell_{mb}^{r_1}, \ell_{mb}^{r_2} - \Delta \ell^{r_1}), \\
\ell_{Me}^r &= \min(\ell_{Me}^{r_2}, \ell_{Me}^{r_1} + \Delta \ell^{r_2}), \\
\Delta \ell^r &= \min(\Delta \ell^{r_1} + \Delta \ell^{r_2}, \ell_{Me}^r - \ell_{mb}^r).
\end{align*}
\]

The proof of Proposition 3 uses the following lemma.

**Lemma 1.** Given a route \( r = z_0, \ldots, z_k \), we have

\[
\ell_{z_k}(\ell^{in}) = \min(\ell_{Me}^r, \ell^{in} + \Delta \ell^r) \quad \text{for any } \ell^{in} \in [\ell_{mb}^r, M^{ch}].
\]

**Proof.** Remark that a simple induction on \( i \) enables to prove that first, \( \ell^{in} \mapsto \ell_{z_i}^r(\ell^{in}) \) is non-decreasing, and second, if there is no station \( \sigma \) in \( z_1, \ldots, z_{i-1} \) such that a vehicle starting with a minimum-level battery has a full battery when leaving \( \sigma \), then \( \ell_{z_i}^r(\ell^{in}) - \ell_{z_i}(\ell^{in}) = \ell^{in} - \ell_{mb}^r \).
Consider the activity \( z_i \) after the first station \( \sigma = z_{i-1} \) such that a vehicle starting with battery level has a full battery when leaving \( \sigma \) if such a station exists. By monotonicity of \( \ell^{in} \mapsto \ell_{z_i}^r(\ell^{in}) \), we obtain \( \ell_{z_i}(\ell^{in}) = \ell_{z_i}(M^{ch}) \), from which we deduce \( \ell_{z_k}^r(\ell^{in}) = \ell_{Me}^r \). This concludes the proof.

**Proof of Proposition 3.** Suppose that \( \ell_{r_1}^r < \ell_{r_2}^r \). Then, a vehicle starting \( r \) with a full battery will arrive in \( \tau \) with battery level \( \ell_{r_1}^r \) and run out of energy while operating \( r_2 \) as \( \ell_{mb}^r \) is the minimum battery level required to operate \( r_2 \) without running out of energy. On the contrary, if \( \ell_{r_1}^r \geq \ell_{r_2}^r \), a vehicle starting \( r \) with a full battery will be able to operate \( r_1 \) and then \( r_2 \) without running out of energy.

Suppose now that \( \ell_{r_1}^r > \ell_{r_2}^r \). If \( \ell_{mb}^r + \Delta \ell^{r_1} > \ell_{mb}^r \), then a vehicle starting \( r_1 \) with battery level \( \ell_{mb}^r \) can operate \( r_2 \) after \( r_1 \). Otherwise, Lemma 1 ensures that the minimum battery level \( \ell^{in} \) at the beginning of \( r_1 \) such that \( \ell_{z_1}^r(\ell^{in}) \geq \ell_{mb}^r \) is \( \ell_{mb}^r - \Delta \ell^{r_1} \), and we obtain \( \ell_{mb}^r = \max(\ell_{mb}^{r_1}, \ell_{mb}^{r_2} - \Delta \ell^{r_1}) \). Starting \( r \) with battery level \( M^{ch} \), a vehicle reaches \( z \) with battery level \( \ell_{Me}^r \), and Lemma 1 ensures that it reaches the end of \( r_2 \) with battery level \( \min(\ell_{Me}^r, \ell_{Me}^r + \Delta \ell^{r_2}) \), which gives \( \ell_{Me}^r = \min(\ell_{Me}^{r_2}, \ell_{Me}^{r_1} + \Delta \ell^{r_2}) \). Lemma 1 ensures that a vehicle starting \( r \) with battery level \( \ell_{mb}^r \) reaches \( z \) with battery level \( \min(\ell_{mb}^r + \Delta \ell^{r_1}, \ell_{Me}^r) \). Lemma 1 again ensures that it reaches the end of \( r_2 \) with charge

\[
\min(\ell_{Me}^r, \ell_{mb}^r + \Delta \ell^{r_1}, \ell_{Me}^r + \Delta \ell^{r_2}) = \min(\ell_{Me}^r, \ell_{mb}^r + \Delta \ell^{r_1} + \Delta \ell^{r_2}, \ell_{Me}^r + \Delta \ell^{r_2})
\]

which gives \( \Delta \ell^r = \min(\Delta \ell^{r_1} + \Delta \ell^{r_2}, \ell_{Me}^r - \ell_{mb}^r) \) and concludes the proof.
3.2.3 Closed Formula for Station Sequences

We now introduce closed formula for the quantities defined in the previous section when \( r \) is a sequence of stations.

**Proposition 4.** Let \( r = z_0, \ldots, z_k \) be a sequence of stations. We define \( e_{z_0}^r \) to be equal to 0 if \( z_0 \in \mathcal{D} \). Let \( e_{nc}^r = \alpha \left( t_{z_k}^b - t_{z_0}^e - \sum_{i=1}^{k} \delta_{z_{i-1},z_i} \right), \) and \( \Delta_{nc}^r = e_{nc}^r - e_{z_0} - \sum_{i=1}^{k} e_{z_{i-1},z_i} \). We have

\[
\ell_{mb}^r = \begin{cases} 0, & \text{if } z_0 \in \mathcal{D}, \\ \max(e_{z_0} + e_{z_0,z_1}, -\Delta_{nc}^r), & \text{if } z_0 \in \mathcal{T}, \end{cases}
\]

\[
\ell_{Me}^r = \min(\mathcal{M}^{ch}, \mathcal{M}^{ch} + e_{nc}^r - e_{z_0} - \sum_{i=1}^{k-1} e_{z_{i-1},z_i}) - e_{z_{k-1},z_k},
\]

\[
\Delta \ell^r = \min(\Delta_{nc}^r, \ell_{Me}^r - \ell_{mb}^r).
\]

**Proof.** \( e_{nc}^r \) is the amount of energy that can be recharged given the time spent in stations, and \( \Delta_{nc}^r \) is the difference between battery level at the end and the beginning if there were no limit on the battery charge. The vehicle is fully charged at the depot, hence \( \ell_{mb}^r = 0 \) if \( z_0 \in \mathcal{D} \). Otherwise, as the sequence is optimally scheduled, a vehicle can run out of energy before reaching the first station, or if it has not enough energy and charge time available to cover the full distance, which gives the value of \( \ell_{mb}^r \). The value of \( \ell_{Me}^r \) follows from Equation (8) in the proof of Proposition 2. The value of \( \Delta \ell^r \) is then a corollary of Lemma 1.

3.2.4 Non-dominated Sequences of Stations

Adler and Mirchandani (2017) note that most sequences of stations are irrelevant. Given two sequences of stations \( r = z_0, \ldots, z_k \) and \( r' = z'_0, \ldots, z'_k \), between the same origin \( z_o = z_o = z'_o \) and destination \( z_d = z_k = z'_k \), sequence \( r \) dominates sequence \( r' \) if

\[
\sum_{i=1}^{k} \delta_{z_{i-1},z_i} \leq \sum_{i=1}^{k'} \delta_{z'_{i-1},z'_i}, \quad \sum_{i=1}^{k} c_{z_{i-1},z_i} \leq \sum_{i=1}^{k'} c_{z'_{i-1},z'_i}, \quad \sum_{i=1}^{k} e_{z_{i-1},z_i} \leq \sum_{i=1}^{k'} e_{z'_{i-1},z'_i}. \tag{9a}
\]

\[
e_{z_0,z_1} \leq e_{z_0,z'_1}, \quad \text{and} \quad e_{z_{k-1},z_k} \leq e_{z'_{k-1},z'_k}. \tag{9b}
\]

For each pair \((z_o, z_d)\) of elements of \( \mathcal{D} \cup \mathcal{T} \), we denote by \( \mathcal{A}_{z_o,z_d} \) a set of sequences of stations between \( z_0 \) and \( z_d \) such that, first, any sequence of stations between \( z_0 \) and \( z_d \) is either in \( \mathcal{A}_{z_o,z_d} \) or dominated by a sequence in \( \mathcal{A}_{z_o,z_d} \), and second, given no sequence in \( \mathcal{A}_{z_o,z_d} \) is dominated by another sequence in \( \mathcal{A}_{z_o,z_d} \). Such an \( \mathcal{A}_{z_o,z_d} \) is easily built using an enumeration algorithm. We denote by \( \mathcal{A} \) the union of the \( \mathcal{A}_{z_o,z_d} \) for \( z_o \) and \( z_d \) in \( \mathcal{D} \cup \mathcal{T} \). We denote by \( \mathcal{R}, \mathcal{R}_c, \) and \( \mathcal{R}_\theta \) the subsets of \( \mathcal{R}, \mathcal{R}_c, \) and \( \mathcal{R}_\theta \) composed of routes whose subroutes are all in \( \mathcal{A} \). The following lemma shows that we can limit the search to routes in \( \mathcal{R} \) when solving the electric vehicle scheduling problem.
Proposition 5. Suppose that \( r \) is a route containing a sequence of station \( s \) dominated by a sequence \( s' \). Then the sequence \( r' \) obtained by replacing \( s \) by \( s' \) is a feasible route and \( c_{r'} \leq c_r \).

Proof. This result is a corollary of the monotonicity of \( \ell_r^t(\ell^m) \) with respect to \( \ell^m \), and of the monotonicity of the closed formula \(-\ell_{mb}^t, \ell_{Me}^t \) and \( \Delta \ell^t \) in Proposition 4 with respect to the quantities in (9).

Our notion of dominance is also more general than the one of Adler and Mirchandani (2017) due to the progressive recharging.

3.3 Reduction of the Pricing Subproblem

We can now reduce our pricing subproblem to the Monoid Resource Constrained Shortest Path Problem.

3.3.1 Digraph

Let \( D = (V, A) \) be the digraph with vertex set \( V = T \cup \{o, d\} \) and

- an arc \( a_r \) from \( o \) to \( \tau \) for each \( \tau \) in \( T \) and \( r \) in \( A_{\theta, \tau} \),
- an arc \( a_r \) from \( \tau_1 \) to \( \tau_2 \) for each \( \tau_1 \neq \tau_2 \) in \( T \) and \( r \) in \( A_{\tau_1, \tau_2} \),
- an arc \( a_r \) from \( \tau \) to \( d \) for each \( \tau \) in \( T \) and \( r \) in \( A_{\tau, \theta} \).

Remark that the existence of a path from \( \tau_1 \) to \( \tau_2 \) in \( D \) implies that \( t_{\tau_1}^e \leq t_{\tau_2}^b \). As \( t_{\tau}^b < t_{\tau}^e \) for any \( \tau \) in \( T \), we obtain that \( D \) is acyclic.

Lemma 2. Let \( z_0, \ldots, z_k \) be the ordered sequence of depots and trips in a route \( r \) of \( R_\theta \), and \( r_i \) the subroute from \( z_{i-1} \) to \( z_i \) for \( i \) in \([k]\). Then \( z_0, \ldots, z_k \) is a path \( P \) in \( D_\theta \). If \( r \) is a complete route in \( R_\theta^C \), then \( P \) is an o-d path in \( D \). Conversely, given a path \( z_0, \ldots, z_k \) in \( D \), concatenating \( z_0, \ldots, z_k \) in this order gives a route, but not necessarily a feasible one.

Proof. The direct part follows immediately from the definitions of \( D, R_\theta \) and \( R_\theta^C \). For the converse, remark that concatenating a route \( r \) arriving in a trip \( \tau \) and a route \( r' \) leaving \( \tau \) gives a route \( r'' \): it suffices to define the \( t_{z}^b, t''_z^e, q_{mb}^t, q_{Me}^t, q_{\Delta}^t \) to be equal to the corresponding times in \( r \) and \( r' \).

3.3.2 Lattice-Ordered Monoid for Battery Charge

We now introduce a lattice-ordered monoid corresponding to the quantities introduced in Section 3.2.2. Let

\[
S = \{ (q_{mb}, q_{Me}, q_{\Delta}) \in \mathbb{R}^3 : \begin{align*}
q_{mb} &\in [0, M^{ch}] \\
q_{Me} &\in [0, M^{ch}] \\
q_{\Delta} &\in [-M^{ch}, M^{ch}] \\
q_{mb} + q_{\Delta} &\geq 0 \\
q_{mb} + q_{\Delta} &\leq q_{Me} \\
M^{ch} + q_{\Delta} &\geq q_{Me}
\end{align*} \} \cup \{\infty\}. \quad (10)
\]
An element \( q \) of \( S \) will decorate a path that corresponds to a route \( r \). It will be equal to \( \infty \) if \( r \) is not feasible, i.e., if a vehicle that operates \( r \) and starts with a full battery runs out of battery before the end of \( r \). Otherwise, it will be equal to \((q_{mb}, q_{Me}, q_{\Delta})\), where \( q_{mb} = \ell_{mb}^r \), \( q_{Me} = \ell_{Me}^r \), and \( q_{\Delta} = \Delta \ell^r \). Inequality \( q_{mb} + q_{\Delta} \geq 0 \) ensures that the battery charge is non-negative at the end, and constraints \( q_{mb} + q_{\Delta} \leq q_{Me} \) and \( M^{ch} + q_{\Delta} \geq q_{Me} \) come from Lemma 1.

Based on Proposition 3, we define the sum operator \( \oplus \) on \( S \) by \( \infty \oplus q = q \oplus \infty = \infty \) for any \( q \) in \( S \) and

\[
(q_{mb}^1, q_{Me}^1, q_{\Delta}^1) \oplus (q_{mb}^2, q_{Me}^2, q_{\Delta}^2) = \begin{cases} \infty & \text{if } q_{Me}^1 < q_{mb}^2, \\ (q_{mb}^1, q_{Me}^1, q_{\Delta}^1) & \text{otherwise}, \end{cases}
\]

where

\[
\begin{align*}
q_{mb}^s &= \max(q_{mb}^1, q_{mb}^2 - q_{\Delta}^1), \\
q_{Me}^s &= \min(q_{Me}^1, q_{mb}^1 + q_{\Delta}^2), \\
q_{\Delta}^s &= \min(q_{\Delta}^1 + q_{\Delta}^2, q_{Me}^1 - q_{mb}^s).
\end{align*}
\]

**Lemma 3.** \((S, \oplus)\) is a monoid with neutral element \((0, M^{ch}, 0)\).

**Proof.** We start by proving that \( S \) is stable by \( \oplus \). It suffices to consider the case \((q_{mb}^1, q_{Me}^1, q_{\Delta}^1) \oplus (q_{mb}^2, q_{Me}^2, q_{\Delta}^2) = (q_{mb}^s, q_{Me}^s, q_{\Delta}^s)\), the other ones being trivial. First, \( q_{mb}^1 \geq 0 \) gives \( q_{mb}^s \geq 0 \), and \( M^{ch} + q_{\Delta}^1 \geq q_{Me}^1 \geq q_{mb}^2 \) together with \( q_{mb}^1 \leq M^{ch} \) gives \( q_{mb}^s \leq M^{ch} \). Second, \( q_{Me}^1 + q_{\Delta}^2 \geq q_{mb}^2 + q_{\Delta}^s \geq 0 \) and \( q_{mb}^2 \geq 0 \) give \( q_{Me}^s \geq 0 \), while \( q_{mb}^2 \leq M^{ch} \) gives \( q_{Me}^s \leq M^{ch} \). Third, \( M^{ch} + q_{\Delta}^1 + q_{\Delta}^2 \geq q_{Me}^1 + q_{\Delta}^s \geq q_{mb}^2 + q_{\Delta}^s \geq 0 \) and \( q_{Me}^1 - q_{mb}^2 \geq -M^{ch} \) gives \( q_{Me}^s \geq -M^{ch} \), while \( q_{Me}^1 - q_{mb}^2 \leq M^{ch} \) gives \( q_{Me}^s \leq M^{ch} \). Fourth, \( q_{mb}^s + q_{\Delta}^1 + q_{\Delta}^2 = q_{mb}^s + q_{\Delta}^1 \geq 0 \) together with \( q_{mb}^s + q_{Me}^s - q_{mb}^s = q_{Me}^s \geq 0 \) gives \( q_{mb}^s + q_{\Delta}^1 \geq q_{Me}^s \geq 0 \). Fifth, \( q_{\Delta}^s = \min(q_{\Delta}^1 + q_{\Delta}^2, q_{Me}^1 - q_{mb}^s) \) gives \( q_{\Delta}^s \leq q_{\Delta}^s \leq q_{\Delta}^s \). Finally, \( M^{ch} + q_{\Delta}^1 + q_{\Delta}^2 \geq q_{Me}^1 + q_{\Delta}^2 \) and \( q_{Me}^s = \min(q_{Me}^1, q_{Me}^2 + q_{\Delta}^2) \) give \( M^{ch}[s] + q_{\Delta}^s \geq q_{\Delta}^s \).

It is immediate that \((0, M^{ch}, 0)\) is the neutral element for \( \oplus \).

We finally prove that \( \oplus \) is associative. Let \( q_1, q_2 \) and \( q_3 \) be three resources in \( S \). If at least one of these resources is equal to \( \infty \), we have that \((q_1 \oplus q_2) \oplus q_3 = q_1 \oplus (q_2 \oplus q_3) = \infty \). Suppose now that \( q_1 = (q_{mb}^1, q_{Me}^1, q_{\Delta}^1), q_2 = (q_{mb}^2, q_{Me}^2, q_{\Delta}^2), \) and \( q_3 = (q_{mb}^3, q_{Me}^3, q_{\Delta}^3) \). If \( q_{mb}^1 < q_{mb}^2, q_{Me}^1 + q_{\Delta}^2 < q_{mb}^3 \) or \( q_{Me}^1 < q_{\Delta}^3 \), applying (11) gives \((q_1 \oplus q_2) \oplus q_3 = q_1 \oplus (q_2 \oplus q_3) = \infty \). Otherwise, we obtain \((q_1 \oplus q_2) \oplus q_3 = q_1 \oplus (q_2 \oplus q_3) = (q_{mb}^s, q_{Me}^s, q_{\Delta}^s)\) with

\[
\begin{align*}
q_{mb}^s &= \max(q_{mb}^1, q_{mb}^2 - q_{\Delta}^1, q_{mb}^3 - q_{\Delta}^2 - q_{\Delta}^1), \\
q_{Me}^s &= \min(q_{Me}^1 + q_{\Delta}^2, q_{mb}^1 + q_{\Delta}^2, q_{mb}^3 + q_{\Delta}^3, q_{Me}^3), \\
q_{\Delta}^s &= \min(q_{\Delta}^1 + q_{\Delta}^2 + q_{\Delta}^3, q_{Me}^1 - q_{mb}^s).
\end{align*}
\]

which gives the associativity of \( \oplus \) and concludes the proof. \(\square\)

Consider the partial order \( \preceq \) on \( S \) defined by \( q \preceq q \) for any \( q \) in \( S \) and

\[
(q_{mb}^1, q_{Me}^1, q_{\Delta}^1) \preceq (q_{mb}^2, q_{Me}^2, q_{\Delta}^2) \quad \text{if} \quad \begin{cases} q_{mb}^1 \leq q_{mb}^2, \\
q_{Me}^1 \geq q_{Me}^2, \\
q_{\Delta}^1 \geq q_{\Delta}^2.\end{cases}
\]

\(13\)
Lemma 4. Partial order $\preceq$ induces a lattice structure on $S$ with meet operator $\land$ defined by $q \land \infty = q$ for any $q$ in $S$ and

$$(q^1_{mb}, q^1_{Me}, q^1_\Delta) \land (q^2_{mb}, q^2_{Me}, q^2_\Delta) = (\min(q^1_{mb}, q^2_{mb}), \max(q^1_{Me}, q^2_{Me}), \max(q^1_\Delta, q^2_\Delta)).$$

Proof. We start by proving the stability of $S$ by $\land$. Let

$$(q^m_{mb}, q^m_{Me}, q^m_\Delta) = (\min(q^1_{mb}, q^2_{mb}), \max(q^1_{Me}, q^2_{Me}), \max(q^1_\Delta, q^2_\Delta)).$$

It is immediate that $q^m_{mb} \in [0, M^{ch}]$, $q^m_{Me} \in [0, M^{ch}]$, and $q^m_\Delta \in [-M^{ch}, M^{ch}]$. Then $q^1_{mb} + \max(q^1_\Delta, q^2_\Delta) \geq q^1_{mb} + q^1_\Delta \geq 0$, and $q^2_{mb} + \max(q^2_\Delta, q^1_\Delta) \geq q^2_{mb} + q^2_\Delta \geq 0$ give $q^m_{mb} + q^m_\Delta \geq 0$. Besides, $\min(q^1_{mb}, q^2_{mb}) + q^1_\Delta \leq q^1_{mb} + q^1_\Delta \leq q^1_{Me}$ and $\min(q^1_{mb}, q^2_{mb}) + q^2_\Delta \leq q^2_{mb} + q^2_\Delta \leq q^2_{Me}$ give $q^m_{mb} + q^m_\Delta \leq q^m_{Me}$. Finally, $M^{ch} + \max(q^1_\Delta, q^2_\Delta) \geq M^{ch} + q^1_\Delta \geq q^1_{Me}$ and $M^{ch} + \max(q^1_\Delta, q^2_\Delta) \geq M^{ch} + q^2_\Delta \geq q^2_{Me}$ give $M^{ch} + q^m_\Delta \geq q^m_{Me}$.

The lattice structure is then an immediate corollary of the fact that $\mathbb{R}^3$ endowed with its component-wise order is a lattice whose meet operator consists in taking the component-wise minimum.

Lemma 5. $\preceq$ is compatible with $\oplus$

Proof. Let $q_1$, $q_2$ and $q_h$ be elements of $S$ such that $q_1 \preceq q_2$. If $q_1$, $q_2$ or $q_h$ is equal to $\infty$, it is immediate that $q_1 \oplus q_h \preceq q_2 \oplus q_h$ and $q_h \oplus q_1 \preceq q_h \oplus q_2$. Suppose now that $q_1 = (q^1_{mb}, q^1_{Me}, q^1_\Delta)$, $q_2 = (q^2_{mb}, q^2_{Me}, q^2_\Delta)$, and $q_h = (q^h_{mb}, q^h_{Me}, q^h_\Delta)$.

Let $q_a = q_1 \oplus q_h$ and $q_b = q_2 \oplus q_h$. We now prove that $q_a \preceq q_b$. If $q_b = \infty$, the result is immediate. Suppose that $q_a = \infty$. This means $q^1_{Me} \preceq q^h_{mb}$, which implies $q^2_{Me} \preceq q^h_{mb}$, and hence $q_a \preceq q_b = \infty$. Suppose now that $q_a = (q^a_{mb}, q^a_{Me}, q^a_\Delta)$ and $q_b = (q^b_{mb}, q^b_{Me}, q^b_\Delta)$. First, $q^1_{mb} \preceq q^2_{mb}$ and $q^1_\Delta \leq q^2_\Delta$ give $q^a_{mb} \preceq q^b_{mb}$ and $q^a_\Delta \preceq q^b_\Delta$. Second, $q^1_{Me} \geq q^2_{Me}$ gives $q^a_{Me} \geq q^b_{Me}$ and $q^a_\Delta \geq q^b_\Delta$. Third, $q^1_\Delta \leq q^2_\Delta$, $q^a_{mb} \preceq q^b_{mb}$ and $q^a_{Me} \geq q^b_{Me}$, and $q^a_\Delta \preceq q^b_\Delta$. We have proved $q_1 \oplus q_h \preceq q_2 \oplus q_h$.

Similar arguments give $q_h \oplus q_1 \preceq q_h \oplus q_2$. □

Lemmas 3 to 5 give the following theorem.

Theorem 1. $(S, \oplus, \preceq)$ is a lattice-ordered monoid.

3.3.3 Full Lattice-Ordered Monoid

$\mathbb{R}$ endowed with its standard sum and order is a lattice-ordered monoid. The lattice-ordered monoid we use is $S \times \mathbb{R}$ endowed with the component-wise sum and order. It is a lattice-ordered monoid as a product of lattice-ordered monoids.
3.3.4 Arc Resources

Let \( a \) be an arc in \( A \), and \( r = z_0, \ldots, z_k \) be the corresponding station sequence. We define the resource of arc \( a \) to be the pair \((q_a, \beta_a)\), where

\[
\beta_a = c_{z_0} + \sum_{i=1}^{k} c_{z_i-1 z_i} \quad \text{where} \quad z_0 = \begin{cases} 
\epsilon_{\text{veh}} - \lambda_{z_0} & \text{if } z_0 \in T, \\
0 & \text{otherwise.}
\end{cases}
\]  

(13)

and \( q_a \) in \( S \) be equal to \((\ell^r_{\text{mb}}, \ell^r_{\text{Me}}, \Delta \ell^r)\), where \( \ell^r_{\text{mb}} \), \( \ell^r_{\text{Me}} \), and \( \Delta \ell^r \) can be computed using the formula in Proposition 4. We use the notation \( \beta_a(\lambda) \) when we want to underline the dependency of \( \beta_a \) on the vector of reduced costs \( \lambda \).

Proposition 6. Let \( P \) be a path in \( D \) and \( r \) the corresponding route according to Lemma 2. We have

\[
\bigoplus_{a \in P} q_a = \begin{cases} 
\infty & \text{if } r \text{ is not feasible,} \\
(\ell^r_{\text{mb}}, \ell^r_{\text{Me}}, \Delta \ell^r) & \text{otherwise.}
\end{cases}
\]

Proof. We prove the result by induction on the number of arcs in \( P \), using the definition of arc resources and Proposition 4 to initiate the induction, and the definition of \( \bigoplus \) and Proposition 3 to propagate the induction.

\]

3.3.5 Cost and Infeasibility Functions, and Reduction of the Pricing Subproblem

To conclude, we define \( \rho : S \times \mathbb{R} \to \{0, 1\} \) and \( c : S \times \mathbb{R} \to \mathbb{R} \) as

\[
\rho(q, \beta) = \begin{cases} 
1 & \text{if } q = \infty, \\
0 & \text{otherwise,}
\end{cases} \quad \text{and} \quad c(q, \beta) = \beta.
\]  

(14)

The following corollary of Proposition 6 concludes the reduction of the pricing subproblem to the monoid resource-constrained shortest path problem.

Proposition 7. Let \( P \) be an o-d path and \( r \) the corresponding route according to Lemma 2, then \( \rho(q_P, \lambda_P) = 0 \) if and only if \( r \) is feasible, and \( c(q_P, \beta_P) = c_r - \sum_{\tau \in r} \lambda_\tau \).

4 Solution Methods

To find an optimal integer solution, we develop a branch-and-bound algorithm, which solves the linear relaxation of Problem (3) at each node by column generation, using the pricing algorithm presented in the previous section to detect columns of negative reduced cost. This leads to a Branch-and-Price (B&P) algorithm.

4.1 Branch-and-Price Algorithm

As seen in previous studies (see, e.g., Martinelli et al. 2011), it is not possible to branch on route variables \( x_r \) since a branch with \( x_r = 0 \) would lead to this variable being generated again.
by the pricing subproblem. For this reason, given a fractional node solution, the algorithm uses different branching rules according to a priority order. The first branching rule checks any fractional number of vehicles leaving a depot. The second is used if there is any fractional traversal from \( \tau_1 \in T \) to \( \tau_2 \in T \). Finally, the last branching rule is used whenever there is any fractional traversal from a trip \( z_1 \in Z \) to another trip \( z_2 \in Z \). Whereas the first and the second branching rules always split the problem using an integer and a binary value, respectively, the last one will use a binary value if \( z_1 \in T \) or \( z_2 \in T \), and an integer value otherwise. Finally, the branching tree is explored following a best-bound strategy.

### 4.2 Graph Sparsification

To further improve the scalability of our solution methods, we rely on graph sparsification strategies to reduce the number of arcs in the digraph \( D = (V, A) \). Our sparsification is done once and for all at the beginning. It exploits a measure of goodness for the arcs and ensures that sufficiently many good incoming arcs and outgoing arcs are preserved for each vertex, therefore giving enough flexibility in the route choices.

Different categories of arcs exist in our problem: some correspond to direct routes without recharging stops, whereas others involve recharging stops. We denote by \( A' \) the sparsified set of arcs and generally distinguish four arc families in the next paragraph. For each arc family and vertex \( v \) in \( T \), we define a score \( \gamma_a \) indicating how good an arc is. Then, for each vertex \( v \) in \( T \) and each family, we add the \( \nu \) arcs of \( \delta^-(v) \) and the \( \nu \) arcs of \( \delta^+(v) \) with minimum \( \gamma_a \) to \( A' \). With this process, an arc can possibly be added twice, once as an incoming arc and once as an outgoing arc. In that case, a single copy is retained.

The families of arcs are defined as follows, where we recall that \( \beta_a(0) \) corresponds to the definition of Equation (13) when the vector of reduced costs \( \lambda \) is equal to 0:

- **Direct arcs without recharge**, coming from a depot of going to a depot, with \( \nu = 2 \) and \( \gamma_a = c(q_a, \beta(0)) \).

- **Arcs with a recharge** coming from a depot of going to a depot, with \( \nu = 2 \) and \( \gamma_a = c(q_a, \beta(0)) \).

- **Arcs between two timetabled trips** \( z_1 \) and \( z_2 \) without recharge, with \( \nu = 15 \) and \( \gamma_a = c(q_a, \beta(0)) + 0.2(t_{z_2} - t_{z_1}) \).

- **Arcs between two timetabled trips with recharge**, with \( \nu = 2 \) and \( \gamma_a = c(q_a, \beta(0)) + 0.1(t_{z_2} - t_{z_1}) + 0.1(L_a) \), where \( L_a \) corresponds to the lost time and is defined as follows. If \( a \) is the arc \( a_r \) corresponding to the sequence of stations \( r = z_0', \ldots, z_{kr}' \) with \( z_1 = z_0' \) and \( z_2 = z_{kr}' \), then \( L_a \) is the time spent waiting at the beginning of \( z_2 \) by a vehicle that would reach the first station with an empty battery and would then recharge as much as possible in each station. Such a vehicle finishes \( z_1 \) with a battery level \( e_{z_1,z_1}' \) that enables only to reach the first station \( z_1' \), and then stays in each station only the time needed to fully charge its battery, i.e., \( \frac{M_{ch}}{a} \) for the first station \( z_1' \) and \( \frac{e_{z_i-1,z_i}}{a} \) for station \( z_i \) with \( i > 1 \).
We therefore have

\[ L_a = \max \left( 0, t_{z_2}^b - \left( t_{z_1}^c + \sum_{i=2}^{k'} \delta_{z_{i-1},z_i} + \left( \frac{M^{\text{ch}} + \sum_{i=2}^{k'-1} e_{z_1,z_2}}{\alpha} \right) \right) \right). \]

Intuitively, an arc is good if it does not require driving an unnecessary distance and does not make the vehicle waste time waiting for a trip. On arcs with recharge, it is necessary to spend some time recharging, leading to the definition of \( L_a \). Some arcs from and to the depot are always maintained to ensure the feasibility of the instances. Since they are the first or last arcs, losing time on these arcs is not an issue.

After sparsification, the number of paths in \( D' = (V,A') \) is orders of magnitude smaller than in the initial graph. This makes the pricing much more efficient and permits us to find high-quality solutions and upper bounds for cases counting several hundreds of timetabled trips.

### 4.3 Diving Heuristics

Diving heuristics have been successfully applied to many complex combinatorial optimization problems (Sadykov et al. 2019). In its most basic version, a diving heuristic consists of a depth-first exploration of the B&P tree. At each search node, the column generation is first completed, and the route variable \( x_r \) with largest value is fixed to 1 (ties are broken randomly). The diving algorithm terminates whenever an integer solution is achieved, or the column generation does not return any feasible solution.

To speed up the solution process, we rely on the sparsification strategy described in the previous section to achieve a fast column generation. Moreover, we adapt the diving approach to fix multiple routes at each search node, prioritizing those with variables \( x_r \) closest to 1. To that end, we consider the routes in non-decreasing order of their associated \( x_r \) values. Each route according to this order is fixed if it does not serve any timetabled trip from a previously fixed route, otherwise it is skipped. This iterative process is pursued until \( n_{\text{DMR}} \) routes have been fixed for this node or all routes have been considered.

The resulting diving algorithm is generally fast, but it behaves greedily since integer columns are permanently fixed. To improve solution quality, we rely on a more sophisticated approach called strong diving (Sadykov et al. 2019). At each search node, this algorithm variant evaluates several possible candidate routes instead of fixing a single one. For each such candidate, it tentatively fixes the associated route and calculates the result of the column generation. The candidate leading to the smallest linear relaxation after fixing is selected, and the search proceeds to the next node. Preliminary experiments revealed that this approach helps drive some of the method’s early decisions, but its systematic application to all search nodes can lead to computational time overheads. Therefore, in the proposed algorithm, strong diving is used only for a fixed number of steps, after which the method follows up with the regular diving heuristic.
5 Experimental Analyses

This section analyses the performance of the proposed methods and the impact of key methodological choices related to the sparsification and diving algorithm. First, we present the results of the B&P algorithm, considering the regular version and its version using sparsification. Then, we focus on the diving heuristics results and draw sensitivity analysis on its key parameters. All tests are conducted on the large benchmark instance set from Wen et al. (2016), containing instances with \{2, 4, 8\} depots, \{4, 8, 16\} recharging stations and \{100, 500\} trips. Each instance is named following the format $D_wS_xC_yz$, where $w$ is the number of depots, $x$ is the number of recharging stations, $y$ is the number of timetabled trips, and $z$ is an index that differentiates each instance.

An important characteristic of this instance set is that the fixed vehicle cost is set to a large value of $c_{veh} = 10,000$. Consequently, vehicle use costs dominate the rest of the objective contribution related to the distance, indirectly implying a hierarchical objective seeking fleet-size minimization in priority followed by driving cost optimization. To conduct fine-grained analyses, we will therefore report the original cost of the solutions along with their fleet size and driving cost.

All algorithms have been developed in C++, using CPLEX 12.8 for the linear programs of the B&P master problem. Note that in the diving algorithm we rebuild the mathematical model after each route fixing, as this progressively reduces the size of the master formulation and improves the overall solution time. We conduct all experiments on a computer with an Intel Core i7-8700K CPU @ 3.70GHz and 64 GB of RAM, running Ubuntu Linux in a single thread. A time limit of six hours (21,600 seconds) was used for each run.

5.1 Exact solution with the Branch-and-Price Algorithm

As discussed in Section 4, we developed two versions of the B&P algorithm. The first solves a pricing subproblem on the complete underlying graph, whereas the second version solves the pricing problem only on the sparsified graph. The results for the B&P algorithm on the instances with 100 and 500 timetabled trips are presented in Tables 1 and 2. The first four columns of Table 1 report for each instance the best results over five runs from the adaptive large neighborhood search (ALNS) of Wen et al. (2016): the upper bound (UB), the number of vehicles used ($V$), the driving cost ($D$), and the average computational time ($T$) for each run in seconds (ALNS runs were performed on an Intel i7-3520M @ 2.9 GHz, which is 1.6× slower than our processor based on Passmark single-thread CPU ratings). The next two groups of five columns give the results of the regular B&P and the B&P with sparsification: the optimal solution found (Opt), the number of vehicles ($V$), the driving cost ($D$), the computational time ($T$) in seconds, and the number of B&B nodes (#). The best solution value for each instance is highlighted in boldface.

As visible in these experiments, the proposed B&P algorithm can solve all 100-trips instances to proven optimality within less than a minute on average. The optimal solutions have 0.53% fewer vehicles and 6.87% less driving cost than the solutions produced by the ALNS on average.
For instance D2_S4_C100_3 in particular, our approach found an optimal solution with one less vehicle for a marginal increase of distance (0.02%).

For these instances, the sparsification strategy permitted to reduce the computational time by 47%, but it also led to a 2.4% increase in the driving cost (the estimated solution deterioration amounts to 0.02% according to the original objective). For instance D2_S4_C100_5, the B&P with sparsification produced the same result as the algorithm without sparsification, in only one third of the computational time. This shows that the proposed sparsification strategy retains useful arcs and can significantly improve scalability.

We now focus our analyses on the large instances with 500 trips. Due to their size, these instances are very challenging and cannot currently be solved to optimality. We will therefore focus the analysis of Table 2 on the lower bounds produced by the B&P algorithm. Thus, the first two groups of four columns present: the root node value (Root), its time in seconds (T), the lower bound at the end of the time limit (LB), and number of B&B nodes (#). The last columns present the percentage difference between the results of the regular B&P and those of the B&P with sparsification.

We now focus our analyses on the large instances with 500 trips. Due to their size, these instances are very challenging and cannot currently be solved to optimality. We will therefore focus the analysis of Table 2 on the lower bounds produced by the B&P algorithm. Thus, the first two groups of four columns present: the root node value (Root), its time in seconds (T), the lower bound at the end of the time limit (LB), and number of B&B nodes (#). The last columns present the percentage difference between the results of the regular B&P and those of the B&P with sparsification.

Table 1: Branch-and-Price results for 100-trips instance set

Table 2: Branch-and-Price results for 500-trips instance set
For two instances out of ten, the standard B&P algorithm could not complete the root node’s solution within the time limit. In contrast, the approach with sparsification concluded the root node for all instances. It is noteworthy that the lower-bound values found with sparsification are within 0.04% of the original bounds and took 13.7× less computational effort on average (considering the subset of instances for which the original algorithm could provide a bound). Whereas the regular B&P took, on average, more than one hour to solve the root node, the B&P with sparsification took less than five minutes. Still, despite this speedup, we observed that the progress of the bounds after the root node was too slow to expect an optimal solution in a reasonable time. Therefore, we turn our attention towards diving heuristics to produce high-quality upper bounds.

5.2 Performance of the Diving Heuristic

The proposed diving heuristics have some parameters that need careful calibration. The “strong diving depth limit” (SDDL) parameter defines the maximum depth up to which the heuristic uses strong diving. The “strong diving maximum solutions” (SDMS) parameter limits the maximum number of solutions that are inspected during strong branching. The “diving maximum routes” (DMR) parameter sets the number of routes that are fixed in a given iteration of regular diving. Finally, the boolean parameter “singleton routes” (SR) controls the pre-generation of singleton routes, i.e., routes with only one customer, before the column generation starts. We noticed improvements with this strategy in our preliminary test, since it effectively avoids infeasible solutions after fixing variables.

Our calibration experiment was conducted on the instances with 500 trips. We explored small integer values for the parameters since our preliminary tests demonstrated that larger values led to excessive computational time amounts. We therefore consider SDDL ∈ {0, 1, 2, 3}, SDMS ∈ {0, 2, 3}, DMR ∈ {1, 2, 3}, SR ∈ {F, T}. SDDL and SDMS = 0 means turning strong diving off (therefore using only regular diving). Each configuration is labeled as a quadruplet (SDDL, SDMS, DMR, SR).

We report the results of the 42 configurations in Figure 2. We registered the number of vehicles and the average driving cost of the solutions of each algorithm configuration and calculate their relative gap (negative values representing improvements) from the best solutions of the ALNS. Therefore, we plot each configuration in a plane with axes representing the percentage gap in terms of driving cost and number of vehicles. The color of each dot represents the computational time of the configuration, with lighter colors indicating slower configurations.

Based on our experiments, two configurations stand out: the one with the best average driving cost and the one with the smallest average number of vehicles. The solution with the best driving cost uses configuration (0, 0, 1, F). This configuration achieves driving-cost and fleet-size values that are 38.65% and 4.66% better than the ALNS solutions, respectively, for an average computational time of 920.3 seconds. This configuration is also the simplest one, as it does not use strong diving, fixes only one route on each iteration, and does not rely on singleton routes. In contrast, the solution with the smallest average number of vehicles uses configuration (2, 2, 1, T). It achieved average improvements of 38.38% and 4.90% in driving cost and
fleet size over the ALNS solutions, for an average computational time of 1083.2 seconds. This configuration exploits all the features discussed in Section 4.3. First, it pre-generates singleton routes. Then, at each iteration, the strong diving considers two alternative routes and selects the best. This strategy is used up to a depth of two, after which the standard diving is used with one route fixed per iteration.

Since the objective value of the EVSP prioritizes fleet-size minimization on these instances, we will provide detailed results for configuration \((2, 2, 1, T)\) as it is the best for this criterion. Tables 3 and 4 show the results for the instances with 100 and 500 trips, respectively. The first four columns of Table 3 present the best ALNS solutions, and the following four columns present the optimal results found by the regular B&P (from Table 1). The eight remaining columns give the results of the diving algorithm and compare them to the ALNS in terms of their upper bound (UB and associated gap), number of vehicles (V and associated gap), driving cost (D and associated gap), computational time (T) in seconds, and number of nodes (#).

![Figure 2: Sensitivity results for Diving Heuristic using 500-trips instance set](image)

Table 3: Results of the diving heuristic for the 100-trips instances

| Instance | ALNS | B&P | Diving Heuristic |
|----------|------|-----|------------------|
|          | UB   | V   | D    | T    | Opt UB | Gap V | D    | Gap T | #    |
| D2_S4_C100_1 | 211775.0 | 21 | 1775.0 | 252 | 211741.0 | 0.01% | 21 | 0.00% | 1788.1 | 0.74% | 4.4 | 16 |
| D2_S4_C100_2 | 182178.0 | 18 | 2178.0 | 890 | 181932.1 | 5.40% | 19 | 5.56% | 2024.3 | -7.06% | 3.8 | 20 |
| D2_S4_C100_3 | 192230.0 | 19 | 2230.0 | 998 | 182231.7 | -5.14% | 18 | -5.26% | 2356.3 | 5.66% | 3.1 | 12 |
| D2_S4_C100_4 | 212231.0 | 21 | 2231.0 | 319 | 212115.7 | -0.03% | 21 | -0.00% | 2177.9 | -2.38% | 6.3 | 7 |
| D2_S4_C100_5 | 181882.0 | 18 | 1882.0 | 381 | 181685.2 | -0.11% | 18 | -0.00% | 1655.2 | -10.46% | 3.8 | 7 |
| D4_S8_C100_1 | 191600.0 | 19 | 1600.0 | 281 | 191470.7 | -0.05% | 19 | -0.00% | 1499.8 | -6.26% | 4.3 | 13 |
| D4_S8_C100_2 | 192097.0 | 19 | 2097.0 | 500 | 191902.5 | -0.06% | 19 | -0.00% | 1985.5 | -5.32% | 4.4 | 12 |
| D4_S8_C100_3 | 191510.0 | 19 | 1510.0 | 325 | 191401.7 | -0.05% | 19 | -0.00% | 1412.3 | -6.47% | 3.8 | 9 |
| D4_S8_C100_4 | 211612.0 | 21 | 1612.0 | 306 | 211408.4 | -0.01% | 21 | -0.00% | 1597.1 | -9.22% | 5.5 | 15 |
| D4_S8_C100_5 | 191704.0 | 19 | 1704.0 | 311 | 191592.5 | -0.04% | 19 | -0.00% | 1628.4 | -4.43% | 4.7 | 14 |

Average | 195881.9 | 19.4 | 1881.9 | 456.3 | 194754.1 | 19.3 | 1754.1 | 46.8 | 195815.5 | -0.01% | 19.4 | 0.03% | 1815.5 | -3.69% | 4.4 | 12.5 |

Table 3: Results of the diving heuristic for the 100-trips instances
As observed in Table 3, the diving heuristic finds solutions of a quality comparable or better than the ALNS, with a similar number of vehicles (one more in the case of instance D2_S4_C100_2, one less for D2_S4_C100_3) and a 3.69% average improvement of driving cost. It is also noteworthy that the diving heuristic is one order of magnitude faster than the ALNS on those medium-scale instances, as it uses 4.4 seconds on average compared to 5 × 456.3 seconds for ALNS.

Compared to the optimal B&P solutions, the number of vehicles and driving costs of the diving heuristic are 0.56% and 3.41% higher on average, but the computational time is 10.6× smaller (5.6× faster compared to the B&P with sparsification). Notably, the diving heuristic found the optimal solution on instance D2_S4_C100_5, and it found the optimal number of vehicles in all cases but D2_S4_C100_2.

Table 4 now presents the results for the 500-trips instance set using a similar format as Table 3. As the optimal solutions are unknown for these instances, we show the best lower bound provided by the B&P, when available.

| Instance         | ALNS | B&P | Diving Heuristic |
|------------------|------|-----|------------------|
|                  | UB   | V   | D    | T   | LB   | Gap | V   | Gap | D   | Gap | T   | #   |
| D4_S8_C500_1    | 878650.0 | 87 | 8650.0 | 1233 | 835222.8 | 835581.4 | -1.90% | 83 | -4.60% | 5581.4 | -35.47% | 549.3 | 77 |
| D4_S8_C500_2    | 940142.0 | 93 | 10142.0 | 1256 | 925500.6 | 926034.3 | -1.50% | 92 | -1.08% | 6034.3 | -40.50% | 334.1 | 80 |
| D4_S8_C500_3    | 859788.0 | 85 | 9788.0 | 1187 | –   | 766242.6 | -10.88% | 76 | -10.59% | 6242.6 | -36.22% | 2472.5 | 67 |
| D4_S8_C500_4    | 870033.0 | 86 | 10033.0 | 1187 | 825646.6 | 826230.3 | -5.03% | 82 | -4.65% | 6230.3 | -37.90% | 1934.3 | 77 |
| D4_S8_C500_5    | 880386.0 | 87 | 10386.0 | 1198 | 786318.8 | 787294.8 | -10.57% | 78 | -10.34% | 7294.8 | -29.76% | 2778.7 | 77 |
| D8_S16_C500_1   | 869530.0 | 86 | 9530.0 | 1255 | 814907.3 | 815278.9 | -6.24% | 81 | -5.81% | 5278.9 | -44.61% | 462.9 | 70 |
| D8_S16_C500_2   | 869282.0 | 86 | 9282.0 | 1259 | 824939.6 | 825389.6 | -5.05% | 82 | -4.65% | 5389.7 | -41.93% | 669.0 | 68 |
| D8_S16_C500_3   | 877456.0 | 87 | 7456.0 | 1178 | 874670.3 | 875060.7 | -0.27% | 87 | 0.00% | 5060.7 | -32.13% | 755.9 | 80 |
| D8_S16_C500_4   | 82538.0 | 82 | 8538.0 | 1189 | 784539.5 | 784832.4 | -5.28% | 78 | -4.88% | 4832.4 | -43.40% | 263.6 | 67 |
| D8_S16_C500_5   | 858816.0 | 85 | 8816.0 | 1183 | –   | 835129.1 | -2.76% | 83 | -2.35% | 5129.1 | -41.82% | 611.6 | 71 |
| Average         | 873262.1 | 86.4 | 9262.1 | 1212.5 | –   | 827707.4 | -5.25% | 82.2 | -4.90% | 5707.4 | -38.38% | 1083.2 | 71.9 |

Table 4: Diving heuristic results for 500-trips instance set

As visible on these experiments, the diving heuristic vastly surpasses the ALNS in terms of solution quality on these large-scale instances, thereby providing a meaningful solution alternative for these challenging cases. We measure an average driving-cost improvement of 38.38%, suggesting that the ALNS was primarily calibrated for fleet-size minimization. In terms of fleet size, the diving heuristic produced solutions with fewer vehicles than the ALNS for 9 out of the 10 instances, corresponding to a 4.90% reduction in the number of vehicles on average.

Noticeably, a comparison with the available lower bounds produced by the regular B&P algorithm (without sparsification) shows that the diving heuristic solutions are no further than 0.06% from the optimal solution cost. With these values, the number of vehicles found by the diving heuristics is very likely to be optimum in all cases. Finally, the heuristic solution took 1083.2 seconds on average, whereas the B&P was interrupted after 21,600 seconds of computational effort without an optimal solution.
6 Conclusions

In this paper, we have introduced efficient branch-and-price and diving algorithms for the EVSP based on a monoid pricing algorithm. The use of this technique brought us two significant advantages. First, this algebraic framework gave us the flexibility needed to model battery charging and sequences of battery recharge stops. Second, the improved bounds permitted an efficient column generation that, in turn, led to high-quality results on large instances. We could find optimal solutions for all 100-trips instances with the B&P algorithm, and solutions no further than 0.06% from the optimum with the diving heuristic for the challenging 500-trips instances. Compared to previous solution approaches, the proposed diving heuristic finds much higher quality solutions in a shorter time.

The research perspectives connected to this work are numerous. First of all, research can be pursued to increase the scalability of the proposed algorithms. An option along this line is to solve the monoid resource-constrained shortest path problems heuristically, possibly by defining a stronger heuristic dominance within the same algebraic framework. Another promising research perspective concerns the integration of machine learning components for critical steps of the method, e.g., for the sparsification step (e.g., as in Joshi et al. 2019) or the branch choices in the B&P and diving algorithms (see, e.g., Gasse et al. 2019, Bengio et al. 2021). Finally, we recommend pursuing the study of pricing algorithms based on monoid structures, considering a broader range of applications to routing and scheduling problems with non-linear extension functions and evaluating their performance in dynamic and stochastic settings.

Acknowledgments

The authors would like to thank Stefan Røpke for his assistance with the instance files and experimental conventions, as well as Victor Abu-Marrul for help on Figure 2. This research was partially supported by CAPES with Finance Code 001, by CNPq under grants 308528/2018-2 and 315361/2020-4, and by FAPERJ under grants E-26/202.790/2019 and E-26/010.002232/2019 in Brazil. This financial support is gratefully acknowledged.

References

Adler, J.D., P.B. Mirchandani. 2017. The vehicle scheduling problem for fleets with alternative-fuel vehicles. Transportation Science 51(2) 441–456.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46(3) 316–329.

Bengio, Y., A. Lodi, A. Prouvost. 2021. Machine learning for combinatorial optimization: A methodological tour d’horizon. European Journal of Operational Research 290(2) 405–421.

Bodin, L., B. Golden, A. Assad, M. Ball. 1983. Routing and scheduling of vehicles and crews: The state of the art. Computers & Operations Research 10(2) 63–211.

Bolívar, M.A., L. Lozano, A.L. Medaglia. 2014. Acceleration strategies for the weight constrained shortest path problem with replenishment. Optimization Letters 8(8) 2155–2172.
Brandstätter, G., C. Gambella, M. Leitner, E. Malaguti, Fi. Masini, J. Puchinger, M. Ruthmair, D. Vigo. 2016. Overview of optimization problems in electric car-sharing system design and management. H. Dawid, K.F. Doerner, G. Feichtinger, P.M. Kort, A. Seidl, eds., Dynamic Perspectives on Managerial Decision Making. Springer International Publishing, 441–471.

Breunig, U., R. Baldacci, R. Hartl, T. Vidal. 2019. The electric two-echelon vehicle routing problem. Computers & Operations Research 103 198–210.

Bunte, S., N Kliewer. 2009. An overview on vehicle scheduling models. Public Transport 1(4) 299–317.

De Jong, S. 2018. London’s iconic black cabs go electric.

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

Desrosiers, J., Y. Dumas, M.M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. M. Ball, T. L. Magnanti, C.L. Monma, G.L. Nemhauser, eds., Network Routing. North-Holland Amsterdam, 35–139.

Dumitrescu, I., N. Boland. 2003. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. Networks 42(3) 135–153.

Dunne, M.J. 2017. China deploys aggressive mandates to take lead in electric vehicles.

Felipe, Á., M.T. Ortuño, G. Righini, G. Tirado. 2014. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. Transportation Research Part E: Logistics and Transportation Review 71 111–128.

Gasse, M., D. Chetelat, N. Ferroni, L. Charlin, A. Lodi. 2019. Exact combinatorial optimization with graph convolutional neural networks. H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché Buc, E. Fox, R. Garnett, eds., Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc.

Haghani, A., M. Banihashemi. 2002. Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints. Transportation Research Part A 36 309–333.

Hiermann, G., R. Hartl, J. Puchinger, T. Vidal. 2019. Routing a mix of conventional, plug-in hybrid, and electric vehicles. European Journal of Operational Research 272(1) 235–248.

Hiermann, G., J. Puchinger, S. Ropke, R.F. Hartl. 2016. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. European Journal of Operational Research 252(3) 995–1018.

Hu, L., J. Dong, Z. Lin, J. Yang. 2018. Analyzing battery electric vehicle feasibility from taxi travel patterns: The case study of New York City, USA. Transportation Research Part C: Emerging Technologies 87 91–104.

Irnich, S., G. Desaulniers. 2005. Shortest path problems with resource constraints. G. Desaulniers, J. Desrosiers, M.M. Solomon, eds., Column Generation. Springer, New York, 33–65.

Joshi, C.K., T. Laurent, X. Bresson. 2019. An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227.

Keskin, M., B. Çatay. 2018. A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. Computers & Operations Research 100 172–188.

Li, J.-Q. 2014. Transit bus scheduling with limited energy. Transportation Science 48(4) 521–539.

Li, Y., C. Zhan, M. de Jong, Z. Lukszo. 2016. Business innovation and government regulation for the promotion of electric vehicle use: lessons from Shenzhen, China. Journal of Cleaner Production 134 371–383.
London Authorities. 2018. Mayor marks key milestone of 100 rapid charging points across London.

Mahmoud, M., R. Garnett, M. Ferguson, P. Kanaroglou. 2016. Electric buses: A review of alternative powertrains. *Renewable and Sustainable Energy Reviews* **62** 673–684.

Martinelli, R., D. Pecin, M. Poggi, H. Longo. 2011. A branch-cut-and-price algorithm for the capacitated arc routing problem. *International Symposium on Experimental Algorithms*. Springer, 315–326.

Montoya, A., C. Guéret, J.E. Mendoza, J.G. Villegas. 2017. The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological* **103** 87–110.

Parmentier, A. 2019. Algorithms for non-linear and stochastic resource constrained shortest paths. *Mathematical Methods of Operations Research* **89** 281–317.

Parmentier, A., F. Meunier. 2020. Aircraft routing and crew pairing: Updated algorithms at Air France. *Omega* **93** 102073.

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

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

Ribeiro, C.C., F. Soumis. 1994. A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research* **42**(1) 41–52.

Sadykov, R., F. Vanderbeck, A. Pessoa, I. Tahiri, E. Uchoa. 2019. Primal heuristics for branch and price: The assets of diving methods. *INFORMS Journal on Computing* **31**(2) 251–267.

Schiffer, M., M. Schneider, G. Walther, G. Laporte. 2019. Vehicle routing and location-routing with intermediate stops: A review. *Transportation Science* **53**(2) 319–343.

Schiffer, M., G. Walther. 2017. The electric location routing problem with time windows and partial recharging. *European Journal of Operational Research* **260**(3) 995–1013.

Scorrano, M., R. Danielis, M. Giansoldati. 2020. Mandating the use of the electric taxis: The case of Florence. *Transportation Research Part A: Policy and Practice* **132** 402–414.

Smith, O.J., N. Boland, H. Waterer. 2012. Solving shortest path problems with a weight constraint and replenishment arcs. *Computers & Operations Research* **39**(5) 964–984.

Tilk, C., A.-K. Rothenbacher, T. Gschwind, S. Irnich. 2017. Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research* **261**(2) 530–539.

van Kooten Niekerk, M.E., J.M. van den Akker, J.A. Hoogeveen. 2017. Scheduling electric vehicles. *Public Transport* **9**(1-2) 155–176.

Vidal, T., G. Laporte, P. Matl. 2020. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research* **286** 401–416.

Wang, Y., Y. Huang, J. Xu, N. Barclay. 2017. Optimal recharging scheduling for urban electric buses: A case study in Davis. *Transportation Research Part E: Logistics and Transportation Review* **100** 115–132.

Wen, M., E. Linde, S. Ropke, P. Mirchandani, A. Larsen. 2016. An adaptive large neighborhood search heuristic for the Electric Vehicle Scheduling Problem. *Computers & Operations Research* **76** 73–83.