UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI INFORMATICA

CORSO DI DOTTORATO IN INFORMATICA

XXXIII ciclo

TESI DI DOTTORATO DI RICERCA

# An algorithm for the optimal routing of electric vehicles

INF/01

RELATORE
Prof. Alberto CESELLI

CORRELATORE
Prof. Giovanni RIGHINI

COORDINATORE DOTTORATO
Prof. Paolo BOLDI

DOTTORANDO
Dario BEZZI
Matr. R11886

A.A. 2019/2020

# Contents

## Abstract

A relevant variation of the classical VRP problem, called the Electric Vehicle Routing Problem (EVRP), aims to properly route Electric Vehicles planning their stops at Recharge Stations. Electric vehicles have the peculiar feature of being operated by batteries of limited capacity, and therefore may need to visit a charging station in between customer visits.

The aim of this thesis is to develop an exact branch-and-price algorithm for solving this problem to optimality. We realistically assume that partial recharges are possible, even with different recharge technologies during the same route, and that more than one recharge technology can be available at any given station.

Existing methodologies for the EVRP focus on constructive heuristics, and very few benchmark instances with proven optimal solutions are available. We improve and integrate methods from the literature, such as bi-directional dynamic programming algorithms for pricing and incompatibility branching rules.

The resulting branch-and-price algorithm is governed by critical parameters whose value can heavily affect the computational performances, although not the optimality guarantee. To overcome this limitation, we design both combinatorial heuristics and data-driven methods to automatically tune these critical parameters, possibly instance-by-instance in an adaptive fashion.

We perform extensive computational experiments to assess the overall performance of our method and a comparison with another exact approach, and report solutions solved to proven optimality in a benchmark dataset.

# 1 Introduction

The Vehicle Routing Problem (VRP), where a fleet of vehicles must deliver goods to customers in a network, is popular since decades for logistic planning. Since the mid-2000s, electric vehicles have gained popularity in several countries, and there has been a growing interest in the integration of environmental aspects in the study of the VRP and its variants. Several nations and local governments have already established measures aimed at increasing the commercial use of electric vehicles, and European Commission is going to restrict the use of the internal combustion engine vehicles in order to incentivize the use of the so called Green Vehicles. Moreover, in 2015 the United Nations proposed an agenda with 17 sustainable development goals to be implemented by all countries by the year 2030; studies regarding Green VRP especially focus on Goals 12 (Responsible consumption and production) and 13 (Climate action) of that document.

## 1.1 Electric VRP variants

The limited battery capacity of electric vehicles requires to visit recharging station during delivery tour, therefore a recharge planning is required to avoid running out of battery. The scientific literature on the EVRP has rapidly developed: in 2015, Pelletier, Jabali and Laporte [32] presented a survey of EVRP variants. More recently, Keskin, Laporte and Catay [19] classified 49 EVRP papers according to fleet composition, objective function terms, presence of multiple recharge technologies and constraints such as capacities and time windows. Some of these variants are briefly summarized here.

Erdoğan and Miller-Hooks [12] formulated the Green VRP (GVRP), in which routes and recharge of electric vehicles are determined simultaneously. The authors used a modified Clarke and Wright savings heuristic and a density-based clustering algorithm to find initial solutions, followed by a postoptimization phase. Another heuristic approach for the GVRP is given in Koç and Karaoglan [23].

Montoya et al. [29] also studied the GVRP. Their objective is to minimize total distance traveled to serve a set of customers considering vehicle range limitations, maximum route duration, and available recharge infrastructure. Service times and refueling times are assumed to be constant, and no time windows are considered (as in [12]). The authors solved the problem by means of a modified multispace sampling heuristic, including an autonomy reparation procedure to ensure route feasibility.

Omidvar and Tavakkoli-Moghaddam [31] have added delivery time windows, customer demands, and vehicle capacity constraints to the GVRP, while also integrating the effect of congestion, discretizing the working day into intervals with different congestion levels, speeds, and travel times (the vehicles can stay at customer locations during congestion periods). The authors proposed a simulated annealing (SA) algorithm and a genetic algorithm to solve the problem.

Schneider, Stenger, and Goeke [38] formulated the Electric VRP with Time Windows (EVRPTW), with customer time windows, demands and capacity constraints. The authors developed a metaheuristic combining variable neighborhood search (VNS) with TS, and proposed benchmark instances for the EVRPTW.

Preis, Frank, and Nachtigall [33] studied the EVRPTW using an energy consumption model (where the objective is to minimize the total energy consumption), taking into account rolling resistance, air resistance, gradient resistance, and energy recuperation. The carried load is also considered in the energy consumed, with charging always done to maximum capacity in a fixed amount of time. They solved the problem using tabu search (TS) and tested their method on varying altitude scenarios. More recently, Montoya et al. [30] formulated the EVRP with nonlinear charging functions (EVRPNL) and presented the MILP formulation.

Other authors have worked on very similar problems: Afroditi et al. [1] considered the same objective function as in [38] assuming a fixed charging time; no resolution method is proposed for solving the problem, but a few promising research directions are outlined. Bruglieri et al. [6] considered a variant of the EVRPTW in which the battery level reached during charging is a decision variable and the objective is to minimize the weighted sum of vehicles used, travel time, waiting time, and recharging time. They solved the problem using variable neighborhood search branching (VNS). Keskin and Çatay [20] worked on the EVRPTW in which partial recharges are allowed when stopping at a recharging station, a linear charging rate is used to determine the associated charging time and the objective is to minimize total traveled distance; they solved the problem using adaptive large neighborhood search (ALNS).

Schiffer and Walther [36] considered a similar problem in the context of location-routing. Sweda et al. [39] studied the optimal recharge policy with given route.

Desaulniers et al. [10] developed exact resolution methods for four variants of the EVRPTW. They propose a set partitioning formulation that requires a set of feasible routes and is solved via branch-and-price-and-cut algorithms. This is one of the few papers in the field to use an exact approach for an EVRP variant, but assumes only one recharge technology to be available in all recharge stations (no recharging costs are considered). More recently, exact MILP formulation for the original GVRP problem have been proposed by Leggieri and Haouari [24] and Andelmin and Bartolini [2]. Bruglieri et al. [7] propose a two-phase solution approach in which a route is modeled by composition of paths, each one handling a subset of customers without intermediate stops at recharge stations.

A more practical variant of the EVRPTW is the vehicle routing and scheduling problem with soft time windows (EVRPSTW), where deliveries are still possible outside the time windows with some penalty costs. Meng and Ma [27] proposed a MILP formulation for this variant and solved it using an ant colony algorithm.

Bruglieri, Mancini and Pisacane [5] recently introduced the GVRP with Capacitated Alternative Fuel Stations (GVRP-CAFS), where only a limited number of fueling pumps are available at any station. The authors proposed two MILP formulations and an exact cutting planes approach for this variant, along with a benchmark set of challenging instances.

In the context of on-demand passenger transportation by means of electric cars, it is possible to define many other EVRP generalizations, such as the Dial-a-Ride Problem (DARP). This problem consists of defining minimum cost routes and schedules for a fleet of vehicles, exiting a common depot and serving a set of customers with given pickup and dropoff locations and corresponding pickup or dropoff times.

The DARP is more challenging than other routing problems due to the need to weight transportation cost and user inconvenience against each other, but its static version share many similiarities with the EVRP. Bongiovanni, Kaspi and Geroliminis [4] proposed a branch-and-cut algorithm for the DARP and some valid inequalities.

## 1.2 The GVRP-MTPR problem

Felipe et al. [14] formulated the Green Vehicle Routing Problem with Multiple Technologies and Partial Recharges (GVRP-MTPR), a VRP variant where one resource (energy) is renewable. Partial recharges and overnight depot charging are allowed. The fleet consists of homogeneous electric vehicles, and charging implies a fixed stopping time and a variable charging time. A charging speed and cost is associated with each charging technology, and each station has a set of offered technologies, so that faster recharges are possible but more expensive. Therefore, when driving time constraints are imposed, partial charging must be considered as an option allowing a vehicle to trade battery for time, recharging a fraction of its capacity. A fixed charging cost is also used to represent battery aging. The objective is to minimize the total charging costs. The authors proposed a nearest neighbor construction heuristic, and their numerical analyses also demonstrated the cost benefit of considering several charging technologies and allowing partial recharges.

Several other authors proposed heuristic algorithms for this variation of the EVRP: among other, Sassi et al. [35] proposed a MILP formulation and a Local Search heuristic; Li-Ying and Yuan-Bin [25] proposed an adaptive VNS; Koç, Jabali and Laporte [21] proposed a multi-start matheuristic algorithm combining adaptive large neighborhood search and MILP.

More recently, Ceselli et al. [8] presented a branch-and-cut-and-price exact algorithm for the EVRP with multiple technologies. To the best of our knowledge, their algorithm is the only exact method presented so far for this specific problem.

This work is focused on the GVRP-MTPR problem (under the name of EVRP), for which we present an exact algorithm and compare it to the algorithm in [8]. Our approach is based on the branch-and-price framework, and it includes an automated tuning procedure for the most critical parameters.

## 1.3 Main contributions

We remark that only one exact approach (in [8]) has been applied to this problem so far, and it provides significantly weaker lower bounds than ours (see Subsection 7.2 for a comparison). In this thesis:

- We present some properties of the EVRP with multiple technologies;
- We exploit such properties and we develop a branch-and-price algorithm to solve the EVRP;
- We perform extensive computational experiments to both highlight the structural behaviour of our methods and to assess their overall performances;
- We compare our algorithm to the only other exact algorithm studied for the EVRP, and we show that our formulation yields much stronger lower bounds;
- We present a benchmark set of EVRP instances solved to proven optimality.

## 1.4 Structure of this thesis

The thesis is organized as follows. In Section 2 we give a formal description of the EVRP and outline some of its properties; in Section 3 we describe the column generation algorithm and the formulation of the pricing sub-problem. In Section 4 we describe the bi-directional dynamic programming algorithm which is used to solve the pricing problem. In Section 5 we discuss the impact of the critical resource selection on computing time and we describe the automatic classification method we have devised and tested for finding it. In Section 6 we describe the dataset and provide some examples of solved instances. In Section 7 we present computational results and comparisons with the other known exact algorithm; conclusions are drawn in Section 8.

Section 5 of this thesis is based on contents presented in a conference with peer-reviewed published proceedings [3].

# 2 The EVRP Problem

The EVRP as introduced in [14] is by far harder than the classical VRP, both because recharge decisions must be taken in addition to routing decisions and because distance minimization is no longer the only optimization criterion but more complex objective functions must be considered.

## 2.1 Problem formulation

The EVRP problem can be formulated as follows.

Let $G = (\mathcal{N} \cup \mathcal{R}, \mathcal{E})$ be a complete weighted undirected graph whose vertex set is the union of a set $\mathcal{N}$ of *customers* and a set $\mathcal{R}$ of *stations*. A distinguished station $R_0 \in \mathcal{R}$ is the depot (where all vehicle routes start and terminate). Furthermore, let $\mathcal{H} = \{0, \ldots, H\}$ be a set of $H$ *recharge technologies*. Every station in $\mathcal{R}$, including the depot, is equipped with exactly one recharge technology $h \in \mathcal{H}$ having recharge speed $\rho_h$ (in reverse form, time unit/energy unit) and recharge unit cost $\gamma_h$. We assume higher speeds (lower recharge unit times) correspond to higher costs. To simplify the notation, we write $\rho_j$ and $\gamma_j$ to indicate the recharge speed and cost of the technology available at station $j \in \mathcal{R}$ and we denote as $\mathcal{R}_h \subseteq \mathcal{R}$ the set of recharge stations with technology $h$ available. We also assume the depot to have $\rho_0 = 0$ and $0 < \gamma_0 \leq \gamma_j \; \forall j \in \mathcal{R}$ (because it is common to recharge electric vehicles during the night at a lower cost). Note that station vertices are allowed to coincide: stations equipped with more than one technology can be modeled simply by splitting each station equipped with $n$ technologies into $n$ separate stations (not connected to each other).

A demand $q_i$ and a service time $s_i$ are associated with each customer vertex $i \in \mathcal{N}$; all customer vertices must be visited and each customer must be visited by a single vehicle (split delivery is not allowed).

The available fleet consists of $K$ identical vehicles with given capacity $Q$ and equipped with batteries of given capacity $B$. Energy consumption and time consumption are assumed to be proportional to the distance traveled: each edge $e \in \mathcal{E}$ requires $d_e$ units of energy and $t_e$ units of time.

While every customer must be visited once, stations (vertices in $\mathcal{R}$) can be visited at any time by each vehicle, even more than once along the same route. During every recharge operation at station $j \in \mathcal{R}$, a vehicle can recharge any amount $\epsilon$ of energy up to the residual battery capacity, consuming $s_j + \epsilon * \rho_j$ time and paying $f + \epsilon * \gamma_j$, where $s_j$ is the fixed service time for station $j$ and $f$ is a fixed cost to be paid once for each recharge.

The duration of each route is required to be within a given limit $T$, representing the duration of drivers' work shifts. As opposed to the classical VRP, the objective to be optimized is not the distance traveled but the overall recharge cost.

**Observation 2.1.1.** The sum of customer demands is a trivial upper bound to the capacity consumption for each route, so we can set $Q = \sum_{i \in \mathcal{N}} q_i$ without loss of generality if the original value of $Q$ is larger.

**Observation 2.1.2.** Since copies of the same stations are not connected to each other, no feasible solution is allowed to include a recharge partially made with different technologies during the same visit at the same station.

**Observation 2.1.3.** Since we assume Euclidean distances on a complete graph, it can never happen (in an optimal solution) to visit a customer without serving it, or to visit a station without recharging. See also observation 2.3.7.

**Observation 2.1.4.** We should expect $H \leq 4$ in any realistic setting; in fact, every instance considered for the EVRP in this work has $H \leq 3$. In other words, we can treat the number of different recharge technologies as a small constant.

**Observation 2.1.5.** This formulation of the EVRP assumes constant energy consumption traversing edges of the network. In a real scenario, vehicle's load, ambient temperature and driving behavior do affect power consumption. However, a constant energy consumption is suitable, for example, in application contexts where the load of the vehicle is nearly negligible in comparison to the curb weight (such as parcel or letter delivery) and speed is exogenously given (for example, city logistics). Indeed, fixed speed profiles and gradients can be integrated into the computation of $d_e$; see [16]. Incorporating speed as a decision variable will strongly increase the complexity of the problem because travel time and battery consumption would became functions of speed leading to trade-offs to be handled in the dynamic programming algorithm.

## 2.2 Routes

**Definition 2.2.1.** We denote a *route* an ordered sequence $r = (R_0, v_1 \dots v_n, R_0)$ of vertices, where $v_i \in (\mathcal{N} \cup \mathcal{R} - \{R_0\})\ \forall i \in 1 \dots n$, including at most once every customer $i \in \mathcal{N}$. Note that the depot $R_0$ appears as starting and ending vertex of every route, but never as an intermediate vertex.

**Definition 2.2.2.** We define a *recharge plan* $\delta = (\delta_1, \delta_2, \dots)$ for a route $r$ a set of energy amounts to be recharged in each station visited by $r$. Namely, $\delta_j$ is the amount of energy recharged at station $j \in r \cap \mathcal{R}$. Note that the same station can appear more than once in the same recharge plan.

**Definition 2.2.3.** For every route $r$, let us denote $E_r \subseteq \mathcal{E}$ the set of edges with both endpoints in $r$; then we can define the following indicators:

- Total lenght (in energy units) $L_r = \sum_{e \in E_r} d_e$
- Total capacity usage $\phi_r = \sum_{i \in r \cap \mathcal{N}} q_i$

**Definition 2.2.4.** We can also define the following indicators for every route $r$ with recharge plan $\delta$:

- Total time consumed $\tau_r = \sum_{e \in E_r} t_e + \sum_{k \in r} s_k + \sum_{j \in r \cap \mathcal{R}} \delta_j \rho_j$
- Total cost $c_r = \sum_{j \in r \cap \mathcal{R}} (f + \delta_j \gamma_j)$

**Definition 2.2.5.** A route $r$ with recharge plan $\delta$ is *feasible* if and only if these three properties hold:

- Capacity constraint: $\phi_r \leq Q$
- Time constraint: $\tau_r \leq T$
- Battery constraint: Battery charge is kept between 0 and $B$ at any time

**Observation 2.2.1.** The capacity constraint for any route $r$ does not depend on the recharge plan $\delta$ (while the other two feasibility constraints do). Note also that a route can be infeasible w.r.t. time or battery constraint as well, even without considering the recharge plan. For example, a route including two consecutive edges $e', e''$ with $d_{e'} + d_{e''} > B$ surely violates the battery constraint (unless the common vertex is a station).

### 2.2.1 Objective

A feasible solution for the EVRP is a set $S$ of feasible routes such that:

- Every route $r \in S$ is a feasible route;
- Every customer vertex is visited exactly once (in one route).

The EVRP asks for the minimization of the overall recharge cost, $\sum_{r \in S} c_r$.

**Observation 2.2.2.** Neglecting travel times in the objective function is somehow unusual for routing problems, but here we assume $T$ to represent the duration of drivers' work shifts instead of introducing time windows in the model. It is well-known that time windows, especially when they are narrow, significantly help reducing the size of the search space and allow computing provably optimal solutions of larger instances with respect to the case with large time windows (or no time windows at all).

## 2.3 Problem properties

Before stating the mathematical model used for branch-and-price, we outline some observations on key properties of the EVRP.

**Observation 2.3.1.** *Assuming that the recharge cost at the depot is smaller than at the other recharge stations, it is always convenient to start from the depot with full battery and to return to the depot with empty battery (unless the total energy consumption along the route is less than $B$).*

From any route not respecting this property, a better one can be obtained either by increasing the initial charge level or by decreasing the final one, hence decreasing the recharge at a station and increasing the recharge at the depot by the same amount, with a net improvement in the total cost. A route $r$ having $L_r \leq B$ does not perform any recharge operation other than the first one at the depot.

**Observation 2.3.2.** *Being the graph $\mathcal{G}$ symmetric, all feasible solutions to the problem are also symmetric: every route can be feasibly traversed in both directions at the same cost using the same recharge plans (Figure 1).*

This can be counter-intuitive, because energy can be consumed only after it has been stored in the battery. However the limited capacity of the battery places an upper bound on the battery charge level, which is equivalent to a zero lower bound for the reverse solution. In other words, the level of battery charge is constrained to be kept between 0 and $B$ at any point in time along both routes, so that energy in one direction corresponds to residual capacity in the opposite direction. If it's possible to travel from the depot to another vertex $v$ arriving with residual energy $l$, then it's possible to travel the other way along the same path, starting with $B$ unit of energy from $v$ and arriving at the depot with $l$ units remaining.

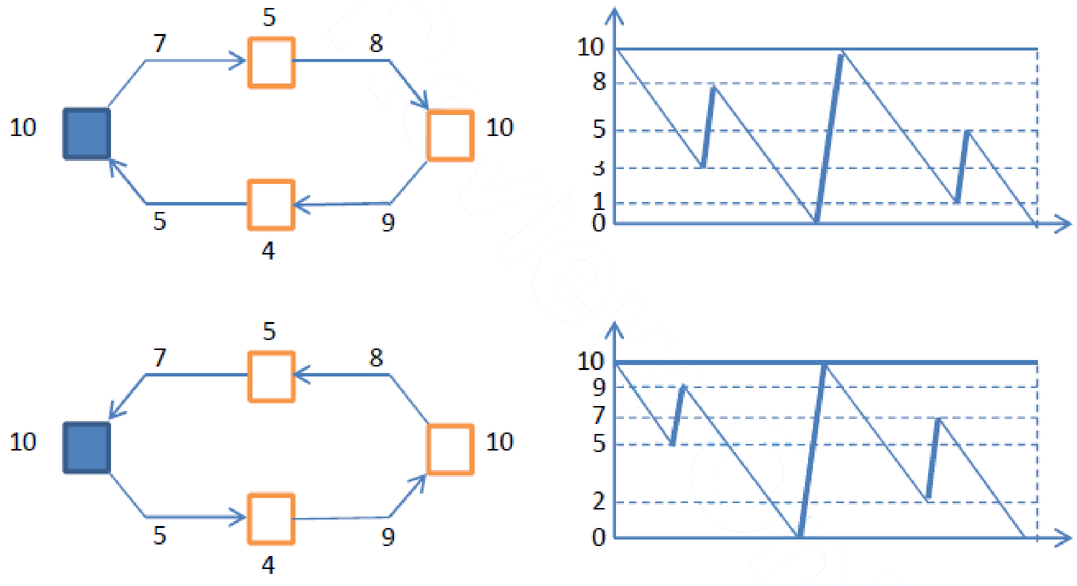


**Figure 1:** Every feasible route remains feasible in both directions.

**Observation 2.3.3.** *Either feasibility or optimality may require a vehicle to visit the same station more than once (Figure 2), or to traverse the same edge more than once (Figure 3).*

In some instances, customers may be far away from the depot, requiring an intermediate step at a recharge station. If that station is the only one (or the most profitable) available in the same area, it will be visited on return as well.

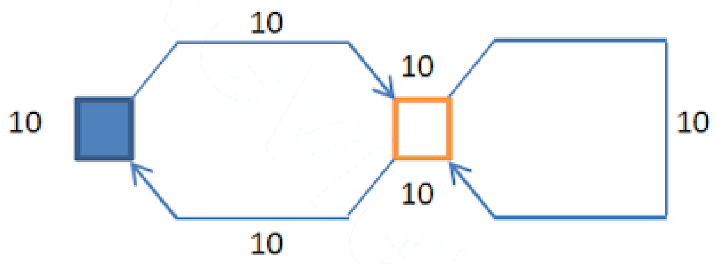


**Figure 2:** Sample instance where feasibility requires multiple visits to the same station.

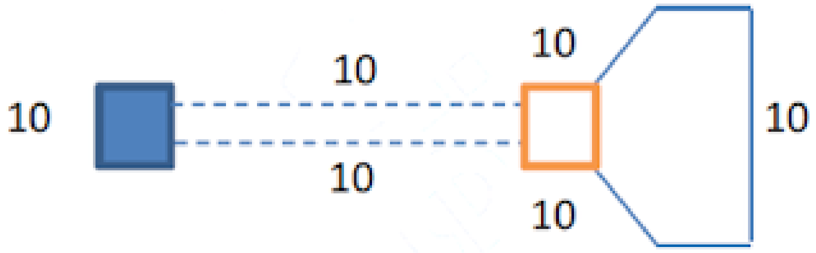


**Figure 3:** Sample instance where feasibility requires multiple traversals of the same edge (dotted lines indicates sets of edges with no customers in between).

**Observation 2.3.4.** *Either feasibility or optimality may require a vehicle to visit two or more stations in a row, without serving any customer in between (figure 3).*

A very "isolated" customer could require to traverse a very long path consuming more than $B$ units of energy, forcing to perform more than one recharge operation without other customers to be served along the way.

**Observation 2.3.5.** *In any feasible solution, at any visit of a recharge station the amount of recharge must be at least equal to the energy consumption along the two adjacent station-to-station paths minus the battery capacity (Figure 4).*

Given a feasible route $r$ and a station $j \in \mathcal{R}$ which is an endpoint for two edges $a, b \in E_r$, it must hold that $\delta_j \geq d_a + d_b - B$ (or the route would violate the battery constraint with a negative battery level between $a$ and $b$). More generally, given a subset of stations $P \subseteq \mathcal{R}$ and defined $Z^1(P)$ as the set of station-to-station subroutes with an endpoint in $P$, then it must hold that $\sum_{j \in P} \delta_j \geq \sum_{l \in Z^1(P)} e_l - B$.

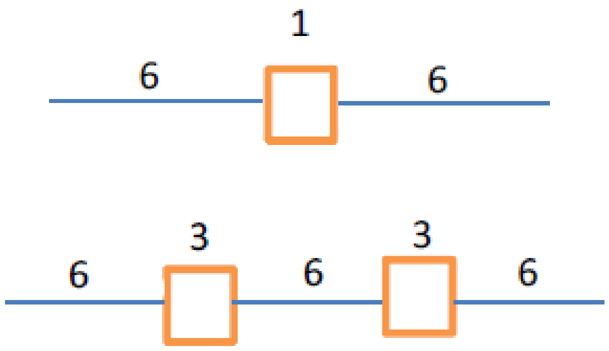


**Figure 4:** Instances violating property 2.3.5.

**Observation 2.3.6.** *In any feasible solution, for any two distinct recharge stations at the endpoints of a path, the overall amount of recharged energy must be at most equal to the energy consumption along the path plus the battery capacity (Figure 5).*

Given a feasible route $r$ and two stations $u, v \in \mathcal{R}$ linked by an edge $a \in E_r$, it must hold that $\delta_u + \delta_v \leq d_a + B$ (or the route would violate the battery constraint with a battery level greater than $B$ at the end of the second recharge operation). More generally, given a subset of stations $P \subseteq \mathcal{R}$ and defined $Z^2(P)$ as the set of station-to-station subroutes with both endpoins in $P$, then it must hold that $\sum_{j \in P} \delta_j \leq \sum_{l \in Z^2(P)} e_l + B$.

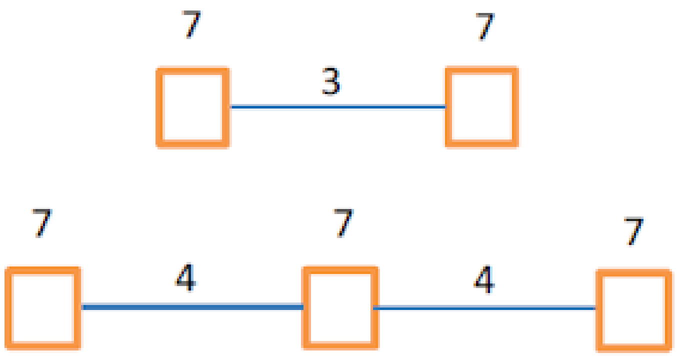


**Figure 5:** Instances violating property 2.3.6.

**Observation 2.3.7.** *For every feasible route $r$ and for every customer $v \in r \cap \mathcal{N}$, another feasible route $r'$ exists such that $r' = r - \{v\}$, and the cost of $r'$ is never higher than the cost of $r$.*

In fact, $r'$ can be built from $r$ simply by shortcutting customer vertex $v$. Being the graph $\mathcal{G}$ complete, the direct edge between the two neighbors of $v$ in the route $r$ surely exist, and its usage cannot violate the battery constraint (since $v$ is a customer and it is not possible to perform recharge operations in customer vertices anyway). Trivially, $r'$ cannot violate neither capacity constraint nor time constraint if $r$ doesn't, so $r'$ must be a feasible route. Moreover, $r'$ cannot consume more energy than $r$, as we assume triangular inequality holds. Thus, the cost of $r'$

cannot be higher than the cost of $r$ (and it will be strictly lower unless $v$ is placed directly on the new edge resulting from shortcutting).

# 3 Model

Route-based formulations are commonly used in the VRP literature: a relevant example in this context is presented in [10], for the electric VRP with time windows. The most common choice to design effective exact optimization algorithms is to rely upon branch-and-price, starting from a reformulation of the routing problem as a set covering or set partitioning problem where each column represents a single route.

We have developed an exact branch-and-price algorithm for the EVRP, relying upon an extended formulation in which each column in the master problem represents a route (see Definition 2.2.1).

## 3.1 Master problem formulation

Let us define $\Omega$ as the set of all feasible routes on the graph $\mathcal{G}$. We associate a binary variable $x_r$ with each feasible route $r \in \Omega$; $x_r$ takes value 1 if and only if the corresponding route $r$ is included in the solution. We also associate binary coefficients $y_{ir}$ with each customer for each route; $y_{ir}$ takes value 1 if and only if customer $i \in \mathcal{N}$ is visited by route $r \in \Omega$. As indicated in Definition 2.2.4, $c_r$ is the total cost of route $r$. With these definitions and notation we obtain the following ILP model as Master Problem ($MP$):

$$\text{minimize} \sum_{r\in\Omega} c_r x_r \tag{1}$$

$$\text{s.t.} \sum_{r\in\Omega} y_{ir} x_r \geq 1 \qquad \forall i \in \mathcal{N} \tag{2}$$

$$\sum_{r\in\Omega} x_r \leq K \tag{3}$$

$$x_r \in \{0,1\} \qquad \forall r \in \Omega \tag{4}$$

The objective function (1) asks for the minimization of the total cost of the selected routes. Covering constraints (2) impose that all customers are visited by at least one route, and route number constraint (3) forbids to include more than $K$ columns in the solution.

At each node of a branch-and-bound tree the linear relaxation of the master problem is solved by column generation. We indicate by $\beta_i$ the dual variables vector corresponding to the covering constraints (2) and by $\mu$ the scalar dual variable corresponding to constraint (3). We assume that all inequality constraints in the $MP$ have been written in $\geq$ form, so that their corresponding dual variables are non-negative. With this notation, the expression of the reduced cost of a generic column $r$ is:

$$\overline{c}_r = c_r - \sum_{r\in\mathcal{N}} \beta_i y_{ir} + \mu$$

**Observation 3.1.1.** Constraints (2) in the $MP$ can be written as set covering constraints even if the EVRP asks for a partition of the customers (no split deliveries are allowed, so each customer must be visited exactly once in each feasible solution and it is trivially sub-optimal for any $x_r$ to take value greater than 1). This is a consequence of Observation 2.3.7. With this formulation the dual space is significantly reduced, as the sign of dual variables is constrained, even if optimal solutions are not affected.

### 3.1.1 Recharge plan for fixed routes

Given a route $r$, finding the recharge plan $\delta^*$ with minimum cost such that $r$ with $\delta^*$ is feasible is indeed a simple task. Since stations are splitted into single-tech vertices, the only decision to be taken for a recharge plan is the amount of energy $\delta_j$ to be purchased at every station $j \in r \cap \mathcal{R}$ (depot included). Moreover, the sum of purchased energy units is trivially equal to $L_r$ in any optimal solution.

We denote by $\underline{\delta}_j$ and $\overline{\delta}_j$ the minimum and maximum amount of feasible recharge for station $j$ in order to respect the battery constraint. This leads to the following LP model (having $\delta_j$ as the only variables):

$$\begin{aligned}
\text{minimize} \quad & \sum_{j \in \delta} \delta_j \gamma_j \\
\text{s.t.} \quad & \sum_{j \in \delta} \delta_j = L_r \\
& \sum_{j \in \delta} \delta_j \rho_j \leq T - \sum_{e \in E_r} t_e - \sum_{k \in r} s_k \\
& \underline{\delta}_j \leq \delta_j \leq \overline{\delta}_j \qquad\qquad \forall j \in \delta
\end{aligned}$$

The optimal recharge plan for a fixed route can be obtained iteratively in linear time, even without solving this simple LP. Therefore, we can assume that knowing only the path for any route $r$ is enough to check feasibility for that route (or to rebuild indicators defined in 2.2.4).

## 3.2 Dummy route

For a strict column generation approach to be performed, a feasible solution for the MP must always exist. For the EVRP, we add a single dummy column in the MP corresponding to a fake route visiting the whole set of customers. The cost of such a dummy column must be higher (possibly just a bit higher) than the overall cost of the optimal solution. In order to set this cost to a proper value, it is possible to search for a real route, feasible w.r.t. battery constraint but disregarding time and capacity constraint. In other words, we compute an actual route $r^*$ visiting all customers, regardless of time and capacity consumption, and set the cost of the dummy column in the MP accordingly (see Observation 3.2.2). We use the greedy approach detailed in algorithm 1 to build $r^*$.

**Observation 3.2.1.** As a consequence of Observation 2.3.7, both $\tau_{r^*}$ and $\phi_{r^*}$ provide bounds to time and capacity consumption for each route. In other words, any single route included in an optimal solution cannot consume more capacity than $\phi_{r^*}$ (already noted in Observation 2.1.1) and cannot spend more time than $\tau_{r^*}$.

Therefore we can set $Q = \phi_{r^*}$ if the original value of $Q$ is larger, and $T = \tau_{r^*}$ if the original value of $T$ is larger (assuming to compute $r^*$ in preprocessing).

**Observation 3.2.2.** If $r^*$ is a feasible route w.r.t. to both time and capacity constraints and its cost is $c_{r^*}$, then $r^*$ alone is a primal solution for the EVRP instance, and the cost of the dummy column in the MP can be directly set to $c_{r^*}$.

In the other case, the cost of the dummy column must be set to an higher value. In our implementation we set it to $Kc_{r^*}$, since it's highly unlikely for a single route belonging to an optimal solution to have a cost greater than $c_{r^*}$ (note, however, that the real cost of the optimal solution can be arbitrarily larger than $c_{r^*}$ for EVRP instances built on purpose).

**Algorithm 1** Greedy algorithm for dummy route

```
$r^* \leftarrow \{R_0\}$
$\rho^{worst} \leftarrow \text{MAX}(\rho_h : h \in \mathcal{H})$    ▷ Slowest tech (with highest recharge unit time)
$\gamma^{worst} \leftarrow \text{MAX}(\gamma_h : h \in \mathcal{H})$    ▷ Most expensive tech
$\phi_{r^*}, \tau_{r^*} \leftarrow 0$
$e \leftarrow B$    ▷ $e$: Residual energy
$c_{r^*} \leftarrow B\gamma_0$
$u \leftarrow R_0$    ▷ $u$: Current vertex
while $(\mathcal{N} \not\subset r^*) \vee (u \neq R_0)$ do
    if $\mathcal{N} \subset r^*$ then
        $i \leftarrow R_0$    ▷ $i$: Target vertex
    else
        $i \leftarrow j \in \mathcal{N} \setminus r^* : d_{[j,u]} = \text{MIN}(d_{[k,u]} : k \in \mathcal{N} \setminus r^*)$    ▷ Nearest customer
    $w \leftarrow j \in \mathcal{R} : d_{[j,i]} = \text{MIN}(d_{[k,i]} : k \in \mathcal{R})$    ▷ Station nearest to $i$
    if $d_{[u,i]} + d_{[i,w]} > e$ then
        $i \leftarrow w$    ▷ Need recharge
    $r^* \leftarrow r^* \cup \{i\}$
    $e \leftarrow e - d_{[u,i]}$
    $\tau_{r^*} \leftarrow \tau_{r^*} + t_{[u,i]} + s_i$
    if $i \in \mathcal{N}$ then
        $\phi_{r^*} \leftarrow \phi_{r^*} + q_i$
    else
        if $i \neq R_0$ then
            $\tau_{r^*} \leftarrow \tau_{r^*} + (B - e)\rho^{worst}$
            $c_{r^*} \leftarrow c_{r^*} + f + (B - e)\gamma^{worst}$
            $e \leftarrow B$
    $u \leftarrow i$
return $(r^*, \phi_{r^*}, \tau_{r^*}, c_{r^*})$
```

### 3.2.1 The Electric TSP problem

It is worth to mention that the subproblem of finding the dummy column with minimum cost can be a nontrivial problem on its own. Informally speaking, let's consider a variation of the classical TSP, including recharge stations and battery constraint as in the EVRP. This problem requires to find the route with minimal cost visiting every customer exactly once (without violating the battery constraint). We call this problem Electric TSP (ETSP); to the best of our knowledge, this specific TSP variant has not yet been described. Note that every EVRP instance can be converted into an ETSP instance by simply dropping time and capacity constraints and allowing only one vehicle to be used. Note also that every optimal solution for an ETSP instance which is feasible w.r.t. original time and capacity constraint is also the optimal solution for the corresponding EVRP instance (this actually happens for some instances in our dataset, see table 14).

## 3.3 Pricing problem formulation

The pricing subproblem is a variation of the Orienteering Problem and it requires to find a minimum cost closed walk from the depot to the depot, not visiting any customer vertex more than once and not consuming more than a given amount of available resources (capacity, time and energy). Edges between stations can be traversed more than once. This problem is also a variation of the Resource Constrained Elementary Shortest Path Problem (which is NP-hard by itself, as stated in [11]), in which the elementary path constraints are imposed only on a subset of vertices, the resources are partly discrete and partly continuous and one of the resources (energy) is renewable.

**Decision variables.** For the ILP formulation of the pricing problem we need:

- A binary variable $z_e$ for each edge $e \in \mathcal{E}$, taking value 1 if and only if the route includes edge $e$;
- A binary variable $y_i$ for each customer $i \in \mathcal{N}$, taking value 1 if and only if the route includes customer $i$;
- An integer variable $\omega_j$ for each station $j \in \mathcal{R}$, indicating the number of visits to station $j$;
- A non-negative, continuous variable $\delta_j$ for each station $j \in \mathcal{R}$, representing the amount of recharge in station $j$.

**Objective function.** The objective function to minimize is the sum of three terms:

- fixed recharge costs, $\sum_{i \in \mathcal{R}} f\omega_i$
- recharge costs, $\sum_{j \in \mathcal{R}} \gamma\delta_j$
- dual prizes, $-\sum_{i \in \mathcal{N}} \beta_i y_{ir} + \mu$

### 3.3.1 Path constraints

We indicate by $\mathcal{E}(j) \subseteq \mathcal{E}$ the set of edges with and endpoint in $j \in \mathcal{N} \cup \mathcal{R}$.

$$\sum_{e \in \mathcal{E}(i)} z_e = 2y_i \qquad \forall i \in \mathcal{N} \tag{5}$$

$$\sum_{e \in \mathcal{E}(j)} z_e = 2\omega_j \qquad \forall j \in \mathcal{R} \tag{6}$$

$$\omega_0 = 1 \tag{7}$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{N} \tag{8}$$

$$\omega_j \text{ integer} \qquad \forall j \in \mathcal{R} \tag{9}$$

$$z_e \in \{0,1\} \qquad \forall e \in \mathcal{E} : i \in \mathcal{N} \vee j \in \mathcal{N} \tag{10}$$

$$z_e \text{ integer} \qquad \forall e \in \mathcal{E} : i \in \mathcal{R} \wedge j \in \mathcal{R} \tag{11}$$

Every feasible route must follow a path which visits at most once each vertex in $\mathcal{N}$ (5), can visit multiple times each vertex in $\mathcal{R}$ (6) and starts at the depot (7). As $y_i$ and $z_e$ are binary variables (8,10), the same customer cannot be served more than once, while $\omega_j$ is an integer, allowing multiple visits to the same station (9). As stated in Observation 2.3.3, it may happen that optimality requires to traverse the same edge more than once; for those edges we treat $z_e$ as an integer (11).

### 3.3.2 Capacity constraints

Route capacity consumption cannot exceed the maximum capacity $Q$:

$$\sum_{i \in \mathcal{N}} q_i y_i \leq Q \tag{12}$$

### 3.3.3 Time constraints

Route duration cannot exceed the maximum time $T$:

$$\sum_{e \in \mathcal{E}} t_e z_e + \sum_{i \in \mathcal{N}} s_i y_i + \sum_{j \in \mathcal{R}} (s_j \omega_j + \rho \delta_j) \leq T \tag{13}$$

Constraint (13) is composed by:

- Total travel time, $\sum_{e \in \mathcal{E}} t_e z_e$;
- Total service time for customers, $\sum_{i \in \mathcal{N}} s_i y_i$;
- Total recharge time in stations, $\sum_{j \in \mathcal{R}} (s_j \omega_j + \rho \delta_j)$.

### 3.3.4 Battery constraints

The maximum amount of energy purchasable from station $j \in \mathcal{R}$ is $B$ times $\omega_i$ (14). Recharges cannot be negative (15).

$$\delta_j \leq B\omega_j \qquad \forall j \in \mathcal{R} \tag{14}$$

$$\delta_j \geq 0 \qquad \forall j \in \mathcal{R} \tag{15}$$

Moreover, let us denote $Z^1(P)$ the set of station-to-station subroutes with an endpoint in $P \subseteq \mathcal{R}$ (as defined in 2.3.5), and $Z^2(P)$ the set of station-to-station subroutes with both endpoints in $P \subseteq \mathcal{R}$ (as defined in 2.3.6).

We assume $\omega_j > 0 \ \forall j \in P$.

$$\sum_{j \in P} \delta_j \geq \sum_{l \in Z^1(P)} \sum_{e \in l} d_e z_e - B \qquad \forall P \subseteq \mathcal{R} \tag{16}$$

$$\sum_{j \in P} \delta_j \leq \sum_{l \in Z^2(P)} \sum_{e \in l} d_e z_e + B \qquad \forall P \subseteq \mathcal{R} \tag{17}$$

Those two constraints force the level of battery charge to be always kept between 0 and $B$.

# 4 Pricing algorithm

The pricing sub-problem is a variant of the Resource Constrained Elementary Shortest Path Problem (RCESPP), that it is known to be NP-Hard [11]. We solve it to optimality by a bi-directional dynamic programming algorithm, where states correspond to partial routes.

## 4.1 Label extension

As in classical VRP algorithms, partial routes are iteratively extended starting from the depot.

### 4.1.1 Feasible Recharge Polyhedron

A *Feasible Recharge Polyhedron* (FRP) $\mathcal{P} = (\underline{\Delta}, \overline{\Delta}, \underline{\delta}, \overline{\delta})$ is defined in as many dimensions as the number of technologies $H$ and it has a special structure:

- vectors $\underline{\delta} = \{\underline{\delta}_0, \dots \underline{\delta}_H\}$ and $\overline{\delta} = \{\overline{\delta}_0, \dots \overline{\delta}_H\}$ represent the minimum and maximum amounts of energy $\delta_h$ recharged for each $h \in \mathcal{H}$;
- the two scalars $\underline{\Delta}$ and $\overline{\Delta}$ represent the minimum and maximum total amount of recharged energy along the path.

The *reference point* of $\mathcal{P}$ is the vertex of $\mathcal{P}$ with minimum cost. We indicate the coordinates of the reference point with a vector $\hat{\delta} = \{\hat{\delta}_0, \dots, \hat{\delta}_{\mathcal{H}}\}$ for any given $\mathcal{P}$. Figure 6 shows an example of a FRP for the case $\mathcal{H} = \{1, 2\}$.

### 4.1.2 State

The Dynamic Programming (DP) label $\mathcal{L} = (u, S, \phi, \mathcal{P}, \tau, c)$ corresponding to a path from the depot to vertex $u$ is defined as follows:

- $u \in \mathcal{N} \cup \mathcal{R}$ is the last vertex reached by the path, that starts from the depot;
- $S \subseteq \mathcal{N}$ is the set of customers visited by the path;
- $\phi$ is the amount of capacity already consumed: $\phi = \sum_{i \in S} q_i$;
- $\mathcal{P}$ is the FRP representing the set of feasible recharge plans;
- $\tau$ is the time needed to reach the *reference point* of $\mathcal{P}$ along the path;
- $c$ is the cost to reach the *reference point* of $\mathcal{P}$ along the path.

Every DP label represents an infinite sets of states (points in the FRP).

**Observation 4.1.1.** The information conveyed by $\tau$ and $c$ is indicated for convenience, but it can be obtained from the knowledge of $\mathcal{P}$.

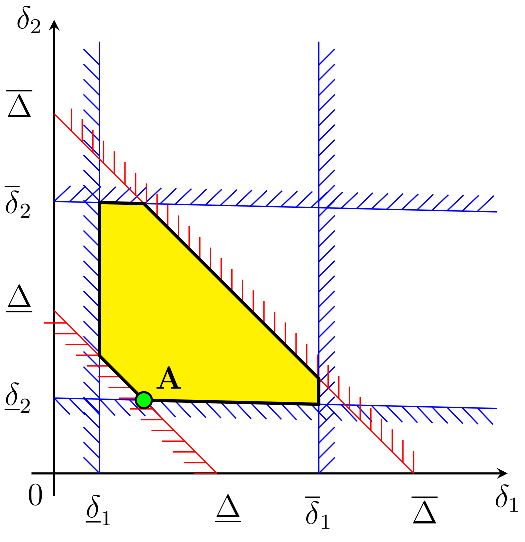


**Figure 6:** The reference point of a feasible recharge polyhedron $\mathcal{P}$ is the vertex of $\mathcal{P}$ with minimum cost (point **A** in the figure). We assume $\gamma_1 < \gamma_2$.

### 4.1.3 Inizialization

The initial state is $\mathcal{L}_0 = (R_0, \emptyset, 0, \mathcal{P}^0, 0, 0)$, corresponding to a partial route including only the depot $R_0$. The polyhedron $\mathcal{P}^0$ is defined by:

$$\mathcal{P}^0 = \begin{cases} \underline{\delta_h} = 0 \ \forall h \in \mathcal{H} \\ \overline{\delta_h} = 0 \ \forall h \in \mathcal{H} \backslash \{0\} \\ \overline{\delta_0} = B \\ \underline{\Delta} = 0 \\ \overline{\Delta} = B \end{cases}$$

### 4.1.4 Feasibility check

For an extension from a label $\mathcal{L}' = (i, S', \phi', \mathcal{P}', \tau', c')$ of vertex $i$ to a label $\mathcal{L}''$ of vertex $j$ along an edge $e = [i, j]$ to be feasible, the following conditions are tested:

- Elementarity: $j \notin S'$
- Capacity: $\phi' + q_i/2 + q_j/2 \leq Q$ (assuming $q_j = 0 \ \forall j \in \mathcal{R}$)
- Duration: $\tau' + s_i/2 + t_e + s_j/2 \leq T$
- Energy: $\overline{\Delta}' - \underline{\Delta}' \geq d_e$

**Observation 4.1.2.** Considering that every feasible route must return to the depot (and it's always convenient to return empty, see Observation 2.3.1), the formulation of duration and energy constraint can be strengthened:

- Duration: $\tau' + s_i/2 + t_e + s_j/2 + t_{[0,j]} \leq T$
- Energy: $\overline{\Delta}' - \underline{\Delta}' \geq d_e + m_j$

Here, $t_{[0,j]}$ is the travel time for the direct edge from $j$ to the depot, while $m_j$ is the amount of energy needed to reach the nearest recharge station from vertex $j$ (it can be precomputed in linear time for each vertex: $m_j = \min_{k\in\mathcal{R}} d_{[j,k]}$).

**Observation 4.1.3.** If $j \in \mathcal{R}$ and no customer vertex has been visited after the last visit to $j$, the extension toward $j$ can be forbidden anyway. Keeping a boolean flag for each station is enough for this test.

**Observation 4.1.4.** If $i, j \in \mathcal{R}_h$ for some $h \in \mathcal{H}$ and $\mathcal{L}'$ was generated by an extension from a third station $k \in \mathcal{R}$ toward $i$, then the direct extension from $k$ to $j$ (if feasible) cannot be worse than the tour $k - i - j$. Therefore the extension from $i$ to $j$ can be disregarded.

### 4.1.5 Label extension

For each label $\mathcal{L}' = (i, S', \phi', \mathcal{P}', \tau', c')$ which does not violate the feasibility checks, we perform the extension phase. $\mathcal{L}'$ is extended from vertex $i$ to every suitable vertex $j$ along edge $e$, and for every $j$ a new label $\mathcal{L}'' = (j, S'', \phi'', \mathcal{P}'', \tau'', c'')$ is created this way:

1. First, we compute $\mathcal{P}''$ and $\hat{\delta}''$ as indicated in Algorithm 2. Traversing an edge $e$ of length $d_e$ increases $\underline{\Delta}$ by $d_e$, which means that $\underline{\delta}_h$ for some $h \in \mathcal{H}$ must increase as well.
2. Then, we perform the following extension rules:
   - $S'' = \begin{cases} S' \cup \{j\} & \text{if } j \in \mathcal{N} \\ S' & \text{if } j \in \mathcal{R} \end{cases}$
   - $\phi'' = \phi' + q_i/2 + q_j/2$ (assuming $q_j = 0\ \forall j \in \mathcal{R}$)
   - $\tau'' = \tau' + s_i/2 + t_e + s_j/2 + \sum_{h\in\mathcal{H}} \rho_h(\hat{\delta}''_h - \hat{\delta}'_h)$
   - $c'' = c' - \beta_i/2 - \beta_j/2 + \sum_{h\in\mathcal{H}} \gamma_h(\hat{\delta}''_h - \hat{\delta}'_h)$

**Algorithm 2** Extension from $\mathcal{P}'$ to $\mathcal{P}''$ along edge $e$ toward vertex $j$

```
$\overline{\Delta}'' \leftarrow \overline{\Delta}'$
$\underline{\Delta}'' \leftarrow \underline{\Delta}' + d_e$
$Z = d_e$
for $h \in \mathcal{H}$ do
    $\overline{\delta}''_h \leftarrow \overline{\delta}'_h$
    $\underline{\delta}''_h \leftarrow \text{MAX}(\underline{\delta}'_h, \underline{\Delta}'' - \sum_{k\in\mathcal{H}:k\neq h} \overline{\delta}'_k)$
    $\epsilon \leftarrow \text{MIN}(Z, \overline{\delta}'_h - \hat{\delta}'_h)$
    $\hat{\delta}''_h \leftarrow \hat{\delta}'_h + \epsilon$
    $Z \leftarrow Z - \epsilon$
if $j \in \mathcal{R}_h$ then
    $\overline{\Delta}'' \leftarrow \underline{\Delta}'' + B$
    $\overline{\delta}''_h \leftarrow \overline{\delta}'_h + (\overline{\Delta}'' - \overline{\Delta}')$
```

The following figures represent the shrinking and enlarging of a sample FRP during the extension phase. When the vehicle travels along an edge, $\underline{\Delta}$ (and possibly $\underline{\delta}_h$ for some $h$) increases, restricting $\mathcal{P}$ (Figure 7). When the vehicle visits a station equipped with technology $h$, $\overline{\Delta}$ and $\overline{\delta}_h$ increase, enlarging $\mathcal{P}$ (Figure 8).

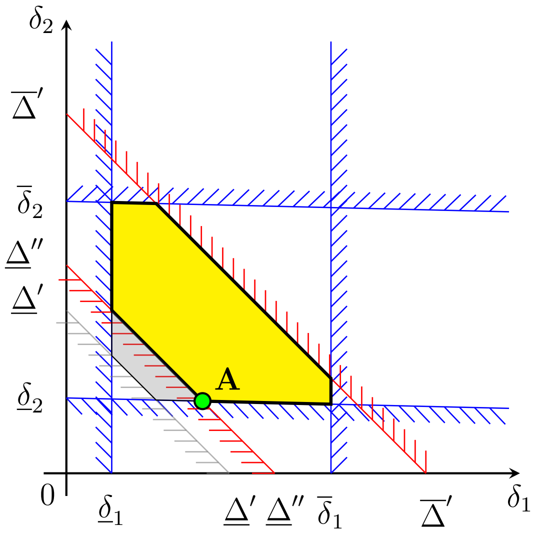


**Figure 7:** Restriction of the FRP when an edge is traversed

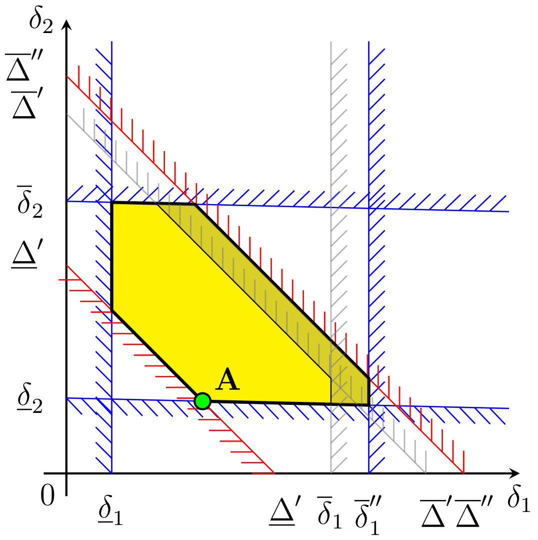


**Figure 8:** Extension of the FRP when a station is visited.

## 4.2 Dominance

The exponential number of labels that have to be potentially considered is a primary source of potential inefficiency. For this reason a dominance test is performed to fathom labels that cannot lead to an optimal solution, thus limiting the combinatorial explosion in the number of labels.

The pricing problem is a variant of the RCESPP, for which efficient dominance rules have been described; see [17]. In our case, a label $\mathcal{L}' = (i, S', \phi', \mathcal{P}', \tau', c')$ dominates a label $\mathcal{L}'' = (i, S'', \phi'', \mathcal{P}'', \tau'', c'')$ if and only if the following inequalities hold and at least one is strict:

- $S' \subseteq S''$
- $\phi' \leq \phi''$
- $\tau' \leq \tau''$
- $c' \leq c''$
- $\mathcal{P}' \supseteq \mathcal{P}''$ when the two reference points are made coincident.

We recall from Subsection 4.1 that $\mathcal{P} = (\underline{\Delta}, \overline{\Delta}, \underline{\delta}, \overline{\delta})$ and $\hat{\delta}$ are the coordinates of the reference point of $\mathcal{P}$. The test for $\mathcal{P}' \supseteq \mathcal{P}''$ requires the following conditions:

- $\overline{\Delta}' - \underline{\Delta}' \geq \overline{\Delta}'' - \underline{\Delta}''$
- $\overline{\delta}'_h - \hat{\delta}'_h \geq \overline{\delta}''_h - \hat{\delta}''_h \quad \forall h \in \mathcal{H}$
- $\hat{\delta}'_h - \underline{\delta}'_h \geq \hat{\delta}''_h - \underline{\delta}''_h \quad \forall h \in \mathcal{H}$

**Observation 4.2.1.** If implemented by bitwise comparison the test $S' \subseteq S''$ takes $O(1)$, implying that every dominance check takes $O(1)$ as a whole.

**Observation 4.2.2.** The dominance test makes sense only when $\mathcal{L}'$ and $\mathcal{L}''$ correspond to the same vertex $i = u' = u''$. For this reason, it is profitable to maintain a set of *buckets* of cardinality $|\mathcal{V}| + |\mathcal{R}|$ (one bucket for each vertex in $\mathcal{G}$) along with the normal queue of DP labels. Each bucket $\lambda_k$ contains (pointers to) all DP labels in the queue corresponding to vertex $k$. This allows to perform dominance checks only between labels belonging to the same bucket.

### 4.2.1 Unreachable customers

The second condition $\phi' \leq \phi''$ is normally implied by the first one, $S' \subseteq S''$. However, if we include in $S'$ and $S''$ also *unreachable customers*, then the second condition must be checked on its own. Unreachable customers are defined as in [13]: in addition to already visited customers, there may be other customers that cannot be visited in any extension of the corresponding path, due to resource limitations. We call such customers vertices unreachable. When a label is extended, visitation resources corresponding to vertices that cannot be visited anymore (either because they have already been visited or because of resource constraints) are consumed. This algorithmic modification is computationally attractive because the dominance relation becomes sharper.

## 4.3 Bi-directional extension

A remarkable speedup in dynamic programming algorithms to compute constrained shortest paths can be achieved by bi-directional extension of labels (see for instance [34]). This allows to generate shorter partial paths with respect to mono-directional extension. The reduction in path length usually pays off, because mono-directional labeling typically suffers from a combinatorial explosion of labels: progressively more labels are created, are extended, and need to be stored when the length (number of edges) of partial paths increases.

In bi-directional dynamic programming, path extension proceeds in two opposite directions, forward and backward, and it is fundamental to stop the extension of forward and backward paths as soon as there is the guarantee that the remaining part of any feasible route can be generated in the opposite direction, so that no feasible solution can be lost. In general, a bi-directional approach requires suitable definitions for:

- backward extension rules;
- extension stop criterion;
- rules for joining semi-routes.

For application where forward and backward labeling differ significantly, such as the EVRP with time windows, it can be worth to define dynamic bounding criterion based on the current state of the simultaneously solved forward and backward RCESPP (see [42]). However, in our case we do not need backward extension rules at all, because no time windows are considered and the graph $\mathcal{G}$ is symmetric (see Observation 2.3.2). Therefore the same partial routes can be traversed in either direction at the same (reduced) cost in the same time.

**Observation 4.3.1.** With bi-directional extension, we can add another forbidden extension (besides those indicated in Observation 4.1.3 and 4.1.4): labels are never extended toward the depot.

### 4.3.1 Extension stop criterion

The extension stop criterion must guarantee that every feasible route can be generated by a pair of partial routes.

**Definition 4.3.1.** We indicate as *critical resource* the resource that is more binding for label extension.

In the EVRP, the pricing sub-problem takes into account three main resources (as defined in Subsection 2.2.5): the amount of demand served, the time needed to travel along the path and the energy consumption along the path. The energy cannot be used as a critical resource, because it can be recharged along a route and therefore routes are not guaranteed to be decomposable in two paths along which no more than half of the battery capacity is consumed. This guarantee holds for time and capacity: so, these are the two resources that can be selected as critical.

We stop the extension when half of the critical resource has been consumed: if an extension would exceed $Q/2$ in capacity consumption (when capacity is considered as critical) or $T/2$ in time consumption (when time is considered as critical), then the extension is not done. This supersedes the feasibility check for label extension stated in Subsection 4.1.4 as follows:

**Critical capacity:**

- Capacity: $\phi' + q_i/2 + q_j/2 \leq Q/2$
- Duration: $\tau' + s_i/2 + t_e + s_j \leq T$

**Critical time:**

- Capacity: $\phi' + q_i/2 + q_j \leq Q$
- Duration: $\tau' + s_i/2 + t_e + s_j/2 \leq T/2$

**Critical resource choice.** The critical resource selection is really important, since a wrong choice would waste (partially or even totally) the advantages of bi-directional extension. Moreover, the critical resource can potentially change according to dual variables associated with customer vertices. However, this happens in practice only for instances built on purpose, and we can assume the critical resource to be the same found for each instance. In theory, the best choice for the critical resource can be performed in preprocessing (thus labelling every instance as "time-critical" or "capacity-critical"), and in the rest of this section we assume the critical resource to be known. For a dedicated algorithm that performs runtime selection of the critical resource, see Section 5.

## 4.4 Join

When two partial routes $\mathcal{L}' = (i, S', \phi', \mathcal{P}', \tau', c')$ and $\mathcal{L}'' = (j, S'', \phi'', \mathcal{P}'', \tau'', c'')$ are joined along the edge $e = [i, j]$, they give origin to a complete route in which the energy consumption is given by $\underline{\Delta}' + \underline{\Delta}'' + d_e$.

The additional cost and time needed to acquire the amount of energy $d_e$ to traverse edge $e$ can be quickly lower bounded by $d_e\rho^*$ (time) and $d_e\gamma^*$ (cost), where $\rho^*$ and $\gamma^*$ are the coefficients of the most convenient technologies among those available in either semi-route (i.e. technologies for which $\hat{\delta}_h < \overline{\delta}_h$). More formally, $\gamma^* = \min_{j \in \mathcal{R} \cap r} \gamma_j$ and $\rho^* = \min_{j \in \mathcal{R} \cap r} \rho_j$.

### 4.4.1 Tech replacement

When the route is generated from two partial routes, traversing it implies traversing one of the two partial routes *to* the depot, instead of *from* the depot.

The energy consumption along the reversed partial route is the same, but the technology used cannot be the same. When partial routes are built, they are built *from* the depot, assuming the possibility of up to $B$ units of energy initially available from technology 0. This is not possible for a reversed partial route, since only one recharge at the depot is allowed to any single route.

**Definition 4.4.1.** After joining two partial routes, some units of energy recharged with technology 0 (available at the depot only) must be replaced using a different technology. In the worst case (routes whose overall consumption is larger or equal to $B$) this implies the replacement of $B$ units of energy.

We call this amount *join energy*:

$$E^{join} = \max(0, \hat{\delta}'_0 + \hat{\delta}''_0 - B)$$

The additional cost and time due to the join energy can be lower bounded by $E^{join}\rho^*$ (time) and $E^{join}\gamma^*$ (cost), as before.

### 4.4.2 Feasibility test

Consider two states $\mathcal{L}' = (i, S', \phi', \mathcal{P}', \tau', c')$ and $\mathcal{L}'' = (j, S'', \phi'', \mathcal{P}'', \tau'', c'')$ and the edge $e = [i, j]$. The following (necessary but not sufficient) conditions are tested to check whether the two partial routes can be joined to produce a feasible route (as stated in Definition 2.2.5):

- Elementarity: $S' \cap S'' = \emptyset$
- Capacity: $\phi' + \phi'' \leq Q - (q_i/2 + q_j/2)$ (assuming $q_j = 0\ \forall j \in \mathcal{R}$)
- Duration: $\tau' + (E^{join} + d_e)\rho^* + \tau'' \leq T - (s_i/2 + t_e + s_j/2)$
- Energy: $(\overline{\Delta}' - \underline{\Delta}') + (\overline{\Delta}'' - \underline{\Delta}'') \geq B + d_e$
- Cost: $c' + (E^{join} + d_e)\gamma^* + c'' \leq \beta_i/2 + \beta_j/2$

### 4.4.3 FRP update

To determine the exact cost and the duration of the resulting route, we need to examine the FRP resulting from joining the two partial routes. From $\mathcal{P}' = (\underline{\Delta}', \overline{\Delta}', \underline{\delta}', \overline{\delta}')$ and $\mathcal{P}'' = (\underline{\Delta}'', \overline{\Delta}'', \underline{\delta}'', \overline{\delta}'')$, we obtain $\mathcal{P}^* = (\underline{\Delta}^*, \overline{\Delta}^*, \underline{\delta}^*, \overline{\delta}^*)$ as follows:

- $\underline{\Delta}^* = \underline{\Delta}' + \underline{\Delta}''$
- $\overline{\Delta}^* = \overline{\Delta}' + \overline{\Delta}''$
- $\overline{\delta}^* = \overline{\delta}' + \overline{\delta}''$
- $\underline{\delta}^* = \underline{\delta}' + \underline{\delta}''$
- $\hat{\delta}^* = \hat{\delta}' + \hat{\delta}''$

However the maximum allowed amount of energy coming from technology 0, i.e. $\overline{\delta}^*_0$, is set to $B$ instead of $\overline{\delta}'_0 + \overline{\delta}''_0 = 2B$.

## 4.5 Optimal recharge plan

After computing $\mathcal{P}^*$, we restrict it and we update the reference point in the same way as after an extension along an edge consuming an amount of energy equal to $d_e$. Then, we must enforce the constraint on the maximum recharge with technology 0, which implies an increase in time and cost for transforming $E^{join}$ units of recharged energy from technology 0 to some other technology.

Let $t$ and $c$ be the initial time and the initial cost of the updated reference point. We transform $E^{join}$ units of energy according to Algorithm 3:

**Algorithm 3** Join: Optimal recharge plan

```
$h \leftarrow 0$
**while** ($\hat{\delta}_0^* > B$) and ($c < 0$) **do**
    $h \leftarrow h + 1$
    $\epsilon \leftarrow \text{MIN}(B - \hat{\delta}_0^*, \overline{\delta}_h^* - \hat{\delta}_h^*)$
    $\hat{\delta}_h^* \leftarrow \hat{\delta}_h^* + \epsilon$
    $\hat{\delta}_0^* \leftarrow \hat{\delta}_0^* - \epsilon$
    $t \leftarrow t + \rho_h \epsilon$
    $c \leftarrow c + \gamma_h \epsilon$
```

When $\hat{\delta}_0^*$ (if initially larger than $B$) goes down to $B$, then a feasible recharge plan of negative cost has been found. When $c$ (initially negative) goes up to 0, no feasible recharge plan with negative cost exists.

### 4.5.1 Repairing infeasible recharge plans

If the updated reference point is not feasible, because the value of $t$ violates the duration constraint, we must determine another point in the polyhedron, trading time for cost. This is done by another replacement of slower recharges with faster recharges, at the expense of increasing the cost.

Finding the minimum cost point in the FRP satisfying the time constraint is a special linear programming problem that can be solved efficiently through Algorithm 4:

**Algorithm 4** Join: Trading time for cost

```
$D \leftarrow \{h \in \mathcal{H} : \hat{\delta}_h > \underline{\delta}_h^*\}$          ▷ Energy can be removed from any tech $h \in D$
$I \leftarrow \{h \in \mathcal{H} : \hat{\delta}_h < \overline{\delta}_h^*\}$               ▷ Energy can be added to any tech $k \in I$
**while** ($t > T$) and ($c < 0$) **do**
    Select $h \in D$ and $k \in I$ such that $\frac{\gamma_k - \gamma_h}{\rho_h - \rho_k}$ is minimum.
    $\epsilon \leftarrow \text{MIN}(\hat{\delta}_h - \underline{\delta}_h^*, \overline{\delta}_k^* - \hat{\delta}_k, \frac{t-T}{\rho_h - \rho_k})$
    $\hat{\delta}_h \leftarrow \hat{\delta}_h - \epsilon$
    $\hat{\delta}_k \leftarrow \hat{\delta}_k + \epsilon$
    $t \leftarrow t - (\rho_h - \rho_k)\epsilon$
    $c \leftarrow c + (\gamma_k - \gamma_h)\epsilon$
```

When $t$ (if initially larger than $T$) goes down to $T$, a feasible recharge plan of negative cost has been found. When $c$ (initially negative) goes up to 0, no feasible recharge plan with negative cost exists.

**Observation 4.5.1.** The definition of Algorithm 4 is necessary in order to guarantee convergence to optimality for the exact pricer, but it rarely matters in practice. If at least another feasible route was already found, the pricer could simply discard the route with the infeasible plan (and possibly raise a warning) instead of trying to fix it. This approach is obviously heuristic, but for all EVRP instances considered in this thesis the optimal solution for the original problem was found anyway, discarding those repairable routes instead of applying Algorithm 4.

## 4.6 Speedup techniques

In order to guarantee convergence to optimality for the MP, the pricing algorithm is required only to return *at least one feasible route with negative reduced cost*, if at least one exists. In practice, it is largely convenient to return feasible routes in batches. However, it can be easily observed that trying to return *all* feasible routes at once is extremely time-consuming, since it requires to perform $O(N^2)$ join operations, being $N$ the number (already exponential) of non-dominated DP labels at the end of the extension phase. The join phase of the pricing algorithm is actually the bottleneck of the entire branch-and-price from a computing time viewpoint, and it is critically important to avoid as many join operations as possible.

### 4.6.1 Avoiding duplicates

The same route can be generated in several different ways. This means that the join operation is executed several times, with no need. To avoid this, we introduce an additional test, that depends on the selected critical resource. That is, the unbalance in the selected resource consumption along the two parts of the resulting route is minimum between $i$ and $j$.

**Critical capacity.** Let $\phi'$ and $\phi''$ the capacity consumptions in vertices $i$ and $j$, respectively. Then $\mathcal{L}'$ and $\mathcal{L}''$ are joined through edge $e = [i, j]$ only if

$$|\phi' - \phi''| \leq (q_i + q_j)/2$$

**Critical time.** Let $\tau'$ and $\tau''$ the time consumptions in vertices $i$ and $j$, respectively, and $\rho^{best}$ the speed of the fastest technology available outside the depot. Then $\mathcal{L}'$ and $\mathcal{L}''$ are joined through edge $e = [i, j]$ only if

$$|\tau' - \tau''| \leq s_i/2 + t_e + \rho^{best} d_e + s_j/2$$

**Observation 4.6.1.** A single join operation requires $O(1)$ time. For this reason, improvements in the join feasibility test cannot really improve the time complexity of a single join: it would consume $O(1)$ time anyway. Duplicate avoidance described here is useful mainly because it forbids to generate many times the same route (avoiding to fill the master solver with useless identical columns).

### 4.6.2 Partial label extension

A quite simple speedup technique consists in a threshold $\mathrm{L}^{max}$ for the number of labels to be generated at each dynamic programming iteration. When the size of label queue reaches $\mathrm{L}^{max}$, the label extension phase freezes and we start to tentatively join labels obtained so far. If no route are found this way, we raise the value of $\mathrm{L}^{max}$ until it reaches the size of label queue.

In our implementation we set $\mathrm{L}^{max} = 200i^3$ for the $i$-th extension iteration.

### 4.6.3 Maximum route number

Another threshold $\mathrm{R}^{max}$ can be set for the number of routes to be returned by the pricer. When the number of already generated feasible routes reaches $\mathrm{R}^{max}$ the whole pricer stops, regardless of how many join operations may still be performed (convergence to optimality is guaranteed anyway). This is particularly convenient if we start to join from the "more promising" partial routes (see 4.6.4 below).

In our implementation we set $\mathrm{R}^{max} = 1000$.

### 4.6.4 Cost-driven join

The list of buckets introduced in Subsection 4.2.2 is useful also during the join phase, if every bucket is sorted by increasing label cost: the first element of $\lambda_k$ is the label $\mathcal{L} = (k, S, \phi, \mathcal{P}, \tau, c)$ with minimum $c$ among all labels ending at $k$, and so on. Intuitively, the more promising pairs of labels $\mathcal{L}'$ and $\mathcal{L}''$ are the ones having minimum values of $c' + c''$, and the more promising edges to use as join edges are the shortest ones. This suggests to perform join operations according to Algorithm 5.

**Discarding buckets.** We indicate as $\lambda_u[i].c$ the (reduced) cost of $i$-th element in bucket $\lambda_u$. Since the join operation requires the whole route to have negative cost, if $\lambda_u[i].c + \lambda_v[j].c + c_{[u,v]}$ is greater or equal to 0, we can immediately conclude that it's pointless to perform a join operation on the corresponding labels (being $c_{[u,v]}$ a lower bound to the cost of the join edge).

We recall that buckets are sorted by increasing cost. Therefore, $\lambda_u[0]$ is the "best" label (the one with larger negative cost) among those ending in vertex $u$. Given the sorted list $\lambda_v$ of labels ending in vertex $v$, if $\lambda_u[i].c + \lambda_v[j].c + c_{[u,v]} \geq 0$ then every label in $\lambda_v[k]$ having $k \geq j$ can be discarded.

**Algorithm 5** Cost-driven join

```
$i \leftarrow 0$
$\mathcal{E}^{sorted} \leftarrow \textsc{sort}(\mathcal{E})$        ▷ Edge set $\mathcal{E}$, sorted by non-increasing length
for $e \in \mathcal{E}^{sorted}$ do
    $[u, v] \leftarrow e$
    $l_1, l_2 \leftarrow 0$
    $c_e \leftarrow d_e\gamma^* - \beta_u/2 - \beta_v/2 - \eta - \mu$        ▷ Edge cost
    while $\lambda_u[0].c + \lambda_v[l_1].c + c_e < 0$ do
        $l_1 \leftarrow l_1 + 1$
    while $\lambda_v[0].c + \lambda_u[l_2].c + c_e < 0$ do
        $l_2 \leftarrow l_2 + 1$
    if $i < s_{max}$ then
        for $j = 0, j < l_1; j = j + 1$ do
            for $k = 0, k < l_2; k = k + 1$ do
                $route \leftarrow \textsc{join}(\lambda_u[j], \lambda_v[k])$
                if $route$ is feasible then
                    Add $route$ to MP
                    $i \leftarrow i + 1$
```

## 4.7 Branching

When a fractional solution arises at the end of the root node, we use the following branching policies to recover integrality. They are considered in the order presented thereafter and branching is executed using the first applicable rule.

### 4.7.1 Branching on route number

*The overall number of routes in the solution is upper bounded by $m$ in one branch and lower bounded by $m+1$ in the other, where $m$ is the integer part of the overall number of routes used in the fractional LP solution.*

This branching rule is applied first because our EVRP instances have very small values for the maximum number $K$ of available vehicles ($2 \leq K \leq 7$ for all instances). In practice, every optimal solution that does not require $K$ vehicles requires at least $K - 1$ vehicles, and a single branch over the route number is enough to determine how many vehicles the instance needs.

### 4.7.2 Branching on customer pairs

*Select a pair of customers $u, v \in \mathcal{N}$ such that the sum of $x_r$ in the $MP$ for every route $r$ including both $u$ and $v$ is closest to* 0.5*. $u$ and $v$ are forced to be visited together (in the same route) in one branch and not allowed to be visited together in the other.*

This rule is the classical branching rule of Ryan-Foster [9], enforced over customer vertices only.

We observe that less than 10 branches on customer pairs largely suffices to reach the global optimum for every instance in our dataset (see Tables 14, 15 and 16).

**Observation 4.7.1.** With the Ryan-Foster rule, if the LP solution is feasible and is integer w.r.t. branching rules 4.7.1 and 4.7.2, then it is the optimal solution of the original problem and no further branching is needed. This is a general property of the Ryan-Foster branching, but it's worthwhile to present a direct proof of this property for our specific problem.

**Proof.** For an EVRP solution to be feasible, every route must be a feasible route by itself, and every customer vertex must be visited exactly once in one route (as stated in 2.2.1). We can assume every column in the MP to already represent a feasible route, as feasibility of single routes w.r.t. capacity, duration and battery constraints is handled by the pricer.

The master solver should only ensure to respect the set partitioning constraints on customer vertices for the routes it choose to include in the solution. As the only nonlinear constraints in the MP are constraints 4 (binary route variables $x_r$), there are two possible sources of non-integrality for the relaxed optimal solution at the root node:

- Non-integer usage of routes
- Split delivery on customers (arising from non-integer route variables)

Note that the optimal relaxed solution cannot have $x_r > 1$ for any $r$, as it is never convenient to pay for the same route more than once.

We recall that not being able to apply branching rule 4.7.2 means that, for every pair of customers $u, v \in \mathcal{N}$, either a single route visits both or they are never visited together. In such a situation, every route variable $x_r$ must take value 0 or 1, because having a route that partially covers $u$ and $v$ would imply the existence of other routes serving the remaining demand of $u$ or $v$ (and it would be possible to branch over pair $[u, v]$).

Finally, note that if one customer is fully served by one route $r$ (with $x_r = 1$), set partitioning constraints for that customer are automatically enforced. In fact, it is never convenient to serve a customer which is already fully served, and for every route $r$ with cost $c$ visiting a customer $v \in \mathcal{N}$, the graph structure implies the existence of another route $r'$ with cost $c' \leq c$ that follows the same path of $r$ except it does not visit $v$ (see Observation 2.3.7). More generally, if $x_r \in \{0, 1\}$ for every $r$ in the LP, then every customer is surely served in full by a single route (and the relaxed optimal solution is also the global optimum for the EVRP instance).

## 5 Automatic selection of the critical resource

Choosing the right critical resource (as defined in 4.3.1) is extremely important in order to preserve the advantage of bi-directional label extension; Subsection 5.1 shows an example of wrong guess consequences.

The task of choosing the right critical resource requires at first a modeling step. In fact, we may define as the *best* critical resource that yielding convergence of the pricing algorithm to an optimal solution in the minimum amount of CPU time. Such a definition, however, can hardly be used in the design of a computational method for finding it.

We therefore explored two options. The first one is to consider the number of labels which are produced by extension operations as a proxy for the overall CPU time required to the pricing algorithm to converge. While the problem of computing the number of labels produced by different choices of the critical resource is as hard as the application of the initial definition, we designed heuristics to provide a reliable estimation of it. Such heuristics still rely on dynamic programming to algorithmically solve a strongly restricted problem at each pricing iteration. We refer to this approach as *partial inspection*, and we detail it in Subsection 5.2.

The second approach is to assume that features on pricing data exist, which are predictive of the best critical resource. Under this assumption we experimented on data-driven methods, modeling the choice of the best critical resource as a supervised learning one. We considered several pricing problem instances arising during EVRP column generation. For each instance we created a data object, measuring and recording several features. We also ran three versions of the pricing algorithm, each one using a different critical resource, recording the CPU time they took to reach proven optimality. Finally, we assigned to the data object a label corresponding to either $T$, $Q$ or $N$, whichever resource corresponds to the faster pricing algorithm version. The set of data objects is then used to train classification models, mapping pricing instance features to critical resource labels. The classification model can then be invoked during the optimization of new EVRP instances to predict at each column generation iteration the best critical resource. We detail this approach in Subsection 5.3.

### 5.1 Critical resource importance

The importance of guessing the right critical resource can be seen from Table 1, where we have reported the number of labels examined and the computing time required by the bi-directional dynamic programming algorithm according to the selection of the critical resource ($T$ stands for time, $Q$ stands for capacity, $N$ stands for number of customers). A wrong guess may lead to an increase in computing time of orders of magnitude. It must be remarked that this guess must be repeated as many times as the number of calls to the pricing sub-routine, which typically occurs some dozen times for each node in the branch-and-cut-and-price tree that in turn may contain up to some hundreds thousands nodes.

From these preliminary results one can appreciate the importance of a reliable and fast technique to automatically select the critical resource in order to keep the overall computing time under control.

**Table 1:** Computing time (sec.) for each resource in single pricing iterations. Note that label extension with non-critical resource can be very time-consuming, i.e. Q and N during the 17th pricer call for instance C103-15.

| Dataset | Instance | Pricer call | $N$ | $Q$ | $T$ |
|---|---|---|---|---|---|
| $A$ | $C101-5$ | 2 | 0.0037 | 0.0021 | 0.0033 |
| $A$ | $C101-10$ | 5 | 4.43 | 3.76 | 0.91 |
| $A$ | $C103-15$ | 1 | 1.07 | 1.11 | 0.61 |
| $A$ | $C103-15$ | 17 | 42.8 | 16.2 | 1.72 |
| $A$ | $C208-15$ | 14 | 5.41 | 5.98 | 7.70 |
| $B$ | $10-N10$ | 1 | 225 | 11.5 | 0.004 |

## 5.2 Selection by partial inspection algorithms

Intuitively, a huge number of label extensions are stopped during each run of the dynamic programming algorithm when the best choice is made for the critical resource. We also expect such a phenomenon to become evident already in the early stages of dynamic programming.

We experimented with heuristics which perform dynamic programming without necessarily performing a complete extension phase. In fact:

- We forbid the extension of each label consuming more than half of *any* resource;
- We stop once a certain number $\mathrm{L}^{max}$ of labels have been generated.

Once this limited dynamic programming is over, we guess the best critical resource to be the one that stopped the highest number of extension operations. A complete pseudo-code of our heuristics is given as Algorithm 6.

Note that, given our choice of RCESPP dynamic programming algorithm, a single extension step could fail even if it does not violate any resource constraint; when it happens, no counter is incremented.

The partial inspection algorithm uses $Q$ as a tiebreaker when all of the critical resources block the same number of extensions, which experimentally only happens when $\overline{N} = \overline{Q} = \overline{T} = 0$. This choice is motivated by the fact that the maximum capacity consumption along a route has a trivial bound (see Observation 2.1.1) while the time constraint has not, and therefore we expect in extreme cases $Q/2$ to be more reliable than $T/2$ as a stopping condition.

## 5.3 Selection by data driven models

The use of heuristics has both appealing and negative features. On one side, they do not require any training, and are ready to be applied to datasets of unknown structure. On the other side, they are time-consuming and might therefore increase the overall computing effort; furthermore, they assume the number of labels which are generated in a limited setting to be a reliable proxy for the computing time of

**Algorithm 6** Partial Inspection Heuristics.

```
$\overline{N}, \overline{Q}, \overline{T}, t \leftarrow 0$
Queue $= \{$empty label$\}$
**while** $t < \mathrm{L}^{\max}$ **do**
    Queue$[t] = \mathcal{L}$
    **if** $\mathcal{L}$ is an open label **then**
        **for** Each node $i$ reachable from $\mathcal{L}$ **do**
            Try to extend $\mathcal{L}$ to $i$ creating $\mathcal{M}$
            **if** $\mathcal{M}$ is blocked **then**
                **if** $\mathcal{M}$ is blocked by $N$ constraint (more than N/2 customers are visited) **then**
                    $\overline{N} \leftarrow \overline{N} + 1$
                **if** $\mathcal{M}$ is blocked by $Q$ constraint (the demand of visited customers exceeds Q/2) **then**
                    $\overline{Q} \leftarrow \overline{Q} + 1$
                **if** $\mathcal{M}$ is blocked by $T$ constraint (the partial route is consuming more that T/2 time) **then**
                    $\overline{T} \leftarrow \overline{T} + 1$
            **else**
                Add $\mathcal{M}$ to Queue
                Perform dominance
    $t = t + 1$
**if** $\overline{N} == \overline{Q} == \overline{T}$ **then**
    **return** $\overline{Q}$
**else**
    **return** max$(\overline{N}, \overline{Q}, \overline{T})$
```

a whole pricing iteration, which may or may not be true. In fact, in preliminary experiments we observed that a few instances exist in which the overall number of labels is not directly correlated with pricing time.

Therefore, we designed the following data driven approach. We assume a dataset of *base instances* to be given. We also assume to perform *simulated runs* on them, performing the full column generation process with *three versions* of the pricing algorithms, one for each possible choice of the critical resource.

For each base instance, and each pricing iteration in its column generation process, we create a data object, which is composed by the following features.

### 5.3.1 Base instance features

We include the number of customers (**N**), vehicles (**K**) and recharge stations (**R**).

### 5.3.2 Mix of base instance features

We include also the following:

**N-normalized** Number of customers divided by number of vehicles: $\tilde{N} = N/K$

**Q-normalized** $\tilde{Q} = \sum_{i \in \mathcal{N}} q_i/Q$

**T-normalized** $\tilde{T} = \sum_{i \in \mathcal{N}} (s_i + d_{i0})/T$ (using depot tech for estimating time)

### 5.3.3 Iteration features

We include the **pricing iteration counter** as a feature as well, which trivially changes over different pricing iterations of the same base instance.

### 5.3.4 Iteration and dual features.

The following values may change over different pricing iterations in non-trivial ways. They are related to the dual solution values. In fact, we argue that the structure of a dual solution might provide insights into the computational behaviour of a pricing algorithm. To give an example, each dual variable having very low value strongly suggests that visiting the associated customer would not pay off; therefore, a high percentage of dual variables having low values suggests optimal routes to be short (thereby making $N$ a bad choice for the critical resource). We recall that $\beta_i$ are the dual variables corresponding to covering constraints in the MP (constraints 2 in Subsection 3.1). In order to obtain promising features, we measure the following:

**Beta-mean** mean value of $\beta_i$

**Beta-variance** variance of $\beta_i$

**Low-N** Percentage of customers having $\beta_i > 0.1 \sum \beta_i$

**Mid-N** Percentage of customers having $\beta_i > 0.2 \sum \beta_i$

**High-N** Percentage of customers having $\beta_i > 0.3 \sum \beta_i$

**Low-Q** Percentage of customers having $q_i\beta_i > 0.1 \sum (q_i\beta_i)$

**Mid-Q** Percentage of customers having $q_i\beta_i > 0.2 \sum (q_i\beta_i)$

**High-Q** Percentage of customers having $q_i\beta_i > 0.3 \sum (q_i\beta_i)$

**Low-T** Percentage of customers having $(s_i + d_{i0})\beta_i > 0.1 \sum (s_i + d_{i0})\beta_i$

**Mid-T** Percentage of customers having $(s_i + d_{i0})\beta_i > 0.2 \sum (s_i + d_{i0})\beta_i$

**High-T** Percentage of customers having $(s_i + d_{i0})\beta_i > 0.3 \sum (s_i + d_{i0})\beta_i$

#### 5.3.5 Class label

Additionally, after all the three pricers are over, we set the following:

**Class** either N, Q or T, depending on which pricer is fastest

## 5.4 Machine learning methods

Then, we exploit the dataset obtained in this way to build classification models with a supervised learning mechanism. That is, we aim to predict class labels (and therefore the best critical resource) by measuring the set of base instance, mix, iteration and iteration dual features. We experimented with many classification models. Indeed, the accuracy is not the only important factor in the choice of the best model, as we may be interested in trading accuracy for ease of embedding in a final branch-and-cut-and-price solver and query efficiency. The following classes were part of our experimental campaign: bayesian classifier, simple decision tree built by information gain induction heuristics and support vector machines with RBF kernel [18]. These models are indeed all easy to directly implement in custom code. As a comparison term, we experimented with a state-of-the-art random forest model trained by gradient boost, although the embedding of the resulting model in custom code (without calling external libraries) is more involved.

## 5.5 Classification results

**Base instance datasets.** As base instance datasets, we used dataset A and B, as described in Section 6. As a preliminary result of our investigation, we found all instances in B to have T (time) as critical resource, for each pricing iteration, so we report detailed results for this dataset only in 5.5.1.

In our experiments we employ a state-of-the-art column generation implementation: the pricing algorithms, as well as the partial inspection heuristics have been implemented in C++; the full column generation algorithm uses SCIP 7.0.1 [15] as a general framework. We set the following time limits:

- 7200 seconds for the full column generation process of each base instance,
- 600 seconds for each single run of a pricing algorithm

- 300 seconds for the label extension phase of every pricing run (that is, we always reserve at least 300s for the join phase).

At each column generation iteration, each pricer might return a set of columns of negative reduced cost (those not hitting the timeout contain an optimal one). In order to keep a reasonable master effort, we keep only at most 100 columns for each pricer (those having minimum reduced cost). That is, at most 300 columns are added to the master problem at each column generation iteration, at least one of which being optimal. We remark that in our experimental setting, each simulation runs three pricing algorithms at each column generation iteration, two of which using a wrong choice of the critical resource, so the overall column generation time in these experiments is not representative of the time required by our procedure in a final implementation (while the time for each pricer does).

The supervised learning models were implemented in R. In particular:

- R Package *e1071* [28] was used to build the bayesian classifier. Function naiveBayes() was called with default parameters;
- R Package *rpart* [40] is used to build the decision tree. Function rpart() is called to construct the tree, and the tree with the minimum prediction error is selected;
- R Package *1071* [28] is used to build the support vector machines classifier, calling function svm() with default parameters;
- R Package *xgboost* [41] is used to build the random forest classifier, calling function xgboost() with parameters max.depth = 3, nrounds = 50 (leaving others as default).

The results were obtained using a PC equipped with an i7-6700K 4.0GHz processor and 32 GB of RAM, running Linux Ubuntu 18.

### 5.5.1 Profiling Partial Inspection Heuristics

**Setup.** The heuristic method described in algorithm 6 is called twice for every pricing iteration, one with a small limit for the number of labels (*fast* heuristic) and one with a significantly larger limit (*slow* heuristic). *Fast* heuristic sets $\mathrm{L}^{max} = 500(N + R)$. *Slow* heuristic sets $\mathrm{L}^{max} = 10000(N + R)$, 20 times more than the *fast* method. The algorithm remains unchanged: labels are tentatively extended until half of one of the resources is consumed, and stopped labels are counted.

In Table 2 we report our results. The table is composed by two blocks, for the slow and fast heuristic respectively. In each block we report the mean ($\pm$ variance) computing time, in seconds (variance values smaller to $10^{-4}$ are rounded to $10^{-4}$), and the accuracy of the heuristic.

One row is included for each of the following sets of base instances:

**A-5** Set of instances having 5 customers belonging to dataset A.

**A-10** Set of instances having 10 customers belonging to dataset A.

**Table 2:** Comparing fast heuristic to slow one

| Set | Slow | | Fast | |
|---|---|---|---|---|
| | Time (s) | Accuracy | Time (s) | Accuracy |
| A-5 | $0.0008 \pm 0.0001$ | 0.96 | $0.0008 \pm 0.0001$ | 0.96 |
| A-10 | $0.45 \pm 3.50$ | 0.66 | $0.080 \pm 0.0037$ | 0.80 |
| A-15 | $24.11 \pm 5339$ | 0.74 | $0.165 \pm 0.010$ | 0.89 |
| B | $0.0009 \pm 0.0001$ | 0.80 | $0.0009 \pm 0.0001$ | 0.80 |

**A-15** Set of instances having 15 customers belonging to dataset A.

**B** Set of all instances belonging to dataset B (having 10 customers).

Each entry represents average values of the instances in the corresponding set.

**Results.** For very small instances (like those in set A-5), there are no differences between the two heuristics. When the instance is so small, the label limit $L^{max}$ is almost never reached even by fast heuristic, so both of them can extend the whole set of labels (and predict the critical resource with high accuracy). A similar behaviour can be observed for dataset B, where time constraint are so strict that halving them leads to drastically reduce feasible extensions, and both heuristics proceed until no open label is available. By converse, the accuracy is not as high as A-5 rows: it is possible to predict a wrong label even with a fully extended queue.

For A-10 and A-15 rows, the most surprising result is that the fast heuristic seems more accurate than the slow one. This particular behaviour suggests again that simply counting the stopped labels is not always an accurate proxy. In details, we found that when $T$ is the critical resource, its effect appears clearly already in early extension operations. At the same time, when this is not the case, the partial inspection heuristics often reach termination before hitting any critical consumption of either $Q$ or $T$. Subsequently, the slow heuristic resorts in predicting $N$, often being wrong. The fast heuristic, instead, according to the tiebreaking condition, correctly predicts $Q$.

### 5.5.2 Classification in synthetic settings

The first experiment is synthetic. It investigates the results obtained by the four classifiers described in Subsection 5.5 in a basic training-and-testing configuration, and compares them to the partial inspection algorithm described in Subsection 5.2.

We randomly split the dataset (325 rows), using 2/3 data objects for training and 1/3 for testing.

That is, representative pricing iterations of each base EVRP instance are expected to be in both training and testing datasets. Each of the four classifiers described in 5.5 split the dataset the same way. Each classifier is permitted to classify according to the features described in Subsection 5.3. In the original dataset, 133 rows are labeled T, 188 are labeled Q and only 4 are labeled N.

**Table 3:** Classification in synthetic settings - training set

| Method | Accuracy | Precision | | | Recall | | |
|---|---|---|---|---|---|---|---|
| | | N | Q | T | N | Q | T |
| Bayesian | 0.78 | 0.10 | 0.87 | 0.92 | 1.00 | 0.75 | 0.82 |
| DTree | 0.93 | 0.00 | 0.95 | 0.93 | - | 0.93 | 0.93 |
| SVM | 0.94 | 0.00 | 0.95 | 0.95 | - | 0.95 | 0.92 |
| Xgboost | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Slow H. | 0.74 | 1.00 | 0.57 | 0.99 | 0.06 | 0.99 | 0.92 |
| Fast H. | 0.88 | 0.67 | 0.84 | 0.94 | 0.13 | 0.95 | 0.92 |

**Table 4:** Classification in synthetic settings - test set

| Method | Accuracy | Precision | | | Recall | | |
|---|---|---|---|---|---|---|---|
| | | N | Q | T | N | Q | T |
| Bayesian | 0.81 | 0.06 | 0.90 | 1.00 | 1.00 | 0.76 | 0.88 |
| DTree | 0.97 | 0.00 | 1.00 | 0.96 | - | 0.95 | 1.00 |
| SVM | 0.95 | 0.00 | 0.98 | 0.94 | - | 0.94 | 0.98 |
| Xgboost | 0.95 | 0.00 | 0.97 | 0.96 | - | 0.95 | 0.98 |
| Slow H. | 0.77 | 0.00 | 0.61 | 0.98 | - | 0.97 | 0.92 |
| Fast H. | 0.85 | 0.00 | 0.86 | 0.85 | - | 0.86 | 0.98 |

The results on the test set are shown in Table 4. Results on the training set are given as a reference in Table 3. Both tables have 6 rows, one for each classification method. Last two rows are included for comparison with partial inspection heuristics. Accuracy (Acc.), Precision (Pr.) and Recall (Rec.) metrics are reported in each table for each classifier, lying in range $[0-1]$. A "-" symbol means that the classifier never predicted class N.

**Results.** A first unexpected result is that classifiers perform in general better than partial inspection heuristics. The bayesian classifier, despite having the worst accuracy, is the only one who correctly predicts N (in testing). We remark that data objects actually classified N are quite rare (in this random split, only 3 for training and 1 for testing). The other three classifiers give similar performances: none of them predicts N, but other data objects are classified with really high accuracy. It is worth noting that Decision Tree and SVM fully discard N: they have Precision 0 for N even during training.

**Table 5:** Features importance in XGBoost.

| Feature | Gain | Cover | Frequence |
|---|---|---|---|
| T-normalized | 0.8066 | 0.163 | 0.132 |
| Beta-variance | 0.0518 | 0.282 | 0.236 |
| Beta-mean | 0.0396 | 0.145 | 0.162 |
| K | 0.0267 | 0.014 | 0.014 |
| N-normalized | 0.0200 | 0.076 | 0.078 |
| Mid-N | 0.0111 | 0.015 | 0.017 |
| Iteration | 0.0107 | 0.067 | 0.064 |
| N | 0.0096 | 0.022 | 0.030 |
| R | 0.0084 | 0.020 | 0.067 |
| Mid-Q | 0.0045 | 0.029 | 0.040 |
| Low-N | 0.0028 | 0.015 | 0.030 |
| Low-Q | 0.0025 | 0.023 | 0.027 |
| High-T | 0.0021 | 0.028 | 0.023 |
| High-N | 0.0017 | 0.052 | 0.030 |
| Mid-T | 0.0009 | 0.024 | 0.017 |
| High-Q | 0.0004 | 0.022 | 0.017 |
| Low-T | 0.0002 | 0.001 | 0.003 |
| Q-normalized | 0.0002 | 0.002 | 0.010 |

**Features importance.** The Decision Tree structure built by R during this experiment is indeed a simple *if* instruction, which classifies according to the **T-normalized** feature only. Under a certain threshold it predicts Q, otherwise it predicts T (data objects with N critical do exist, but they are discarded as outliers). Still, finding a suitable threshold value requires the statistical evaluation of the training set. From the point of view of branch-and-cut-and-price design, this is an appealing phenomenon: it means that base instances could be classified in preprocessing, without any overhead at runtime.

In order to have a more clear picture, in Table 5 we report the relative importance of each feature in the XGBoost model.

The **T-normalized** feature remains the most important one by far, especially in terms of relative gain it produces when used in a node of a tree. At the same time, the mean and variance of dual variables contribute to the efficacy of the method: such a contribution reflects in the values of cover and frequency. No other feature seems relevant.

### 5.5.3 Classification in realistic settings

The second experiment compares classifier performances in a realistic operation mode. That is, we split the dataset in blocks, each containing all and only the rows corresponding to a single base EVRP instance. Experiments are always performed without mixing different blocks during training and testing, thereby simulating a setting in which training is performed on a certain dataset, and then the models are used to perform predictions on new, previously unknown base EVRP instances.

**Table 6:** Classification in realistic settings - training

| Method | Accuracy | Precision | | | Recall | | |
|---|---|---|---|---|---|---|---|
| | | N | Q | T | N | Q | T |
| Bayesian | 0.86 | 0.67 | 0.66 | 0.66 | 0.89 | 0.90 | 0.90 |
| DTree | 0.94 | 0.64 | 0.64 | 0.64 | 0.95 | 0.95 | 0.95 |
| SVM | 0.95 | 0.64 | 0.64 | 0.64 | 0.95 | 0.95 | 0.95 |
| Xgboost | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Slow H. | 0.75 | 0.77 | 0.77 | 0.78 | 0.65 | 0.65 | 0.65 |
| Fast H. | 0.87 | 0.75 | 0.75 | 0.77 | 0.65 | 0.65 | 0.65 |

**Table 7:** Classification in realistic settings - testing

| Method | Accuracy | Precision | | | Recall | | |
|---|---|---|---|---|---|---|---|
| | | N | Q | T | N | Q | T |
| Bayesian | 0.79 | 0.70 | 0.68 | 0.65 | 0.70 | 0.66 | 0.72 |
| DTree | 0.88 | 0.71 | 0.75 | 0.77 | 0.95 | 0.86 | 0.83 |
| SVM | 0.85 | 0.64 | 0.70 | 0.78 | 0.81 | 0.78 | 0.74 |
| Xgboost | 0.92 | 0.67 | 0.73 | 0.86 | 0.81 | 0.87 | 0.92 |
| Slow H. | 0.81 | 0.64 | 0.69 | 0.84 | 0.74 | 0.72 | 0.76 |
| Fast H. | 0.89 | 0.68 | 0.78 | 0.85 | 0.77 | 0.83 | 0.84 |

**Experimental setup.** We employ a leave-one-out approach, iteratively removing one of the blocks described above from the dataset, performing training on the remaining blocks, and testing on the removed one. Since dataset A contains rows corresponding to 36 base EVRP instances, 36 iterations are needed for each classifier, and average results are measured. As before, each classifier is permitted to classify according to the features described in Subsection 5.3.

Our average results on the testing blocks are reported in Table 7. For reference, in Table 6 we report also the average results during training. Both tables have the same structures of those in the previous experiment.

**Results.** This experiment further clarifies the behaviour of the methods. Here, all classifiers are able to predict N (at least for some of the 36 runs). As expected, their mean precision over Q and T decreases. However, it does not drop significantly, especially for Bayesian and XGBoost classifiers. Partial inspection heuristics are competitive in this setting. Indeed, this confirms the intuition that such dedicated heuristics could perform better than general purpose classifiers on a real setting.

As an overall final computational insight, we envisage a cascading approach: at first approximation, a choice of critical resource performed during preprocessing according to base instance features only can be enough. If timeouts are observed during the pricing iterations, very fast queries to an external classifier provide accurate predictions in a large share of cases. Finally, if timeouts are still observed, running partial inspection heuristics can be of further help.

## 5.6 Remarks

Our investigation produced interesting insights. First, partial inspection heuristics provide good accuracy in detecting the best critical resource; in particular, their results are accurate even if very tight bounds are imposed to their computing time and number of labels which are allowed to be processed. In fact, we argue that their biasing towards the best critical resource is more crisp in their earlier phases, tending to a more balanced effect as their computation proceeds.

For what concerns the data driven methods, we have designed a few representative features, and analyzed several classification models from the literature, exploiting their values to predict the best critical resource. Some of these features are related only to the base EVRP instance data; others measure properties of the dual values during a single pricing iteration.

In general, data driven models appear superior to partial inspection heuristics in our experiments.

One feature in particular turns out to be always more significant than the others: it is a (rough) estimate of expected time spent in visiting each customer, expressed in relative terms w.r.t. the time limit $T$. In fact, an unexpected result is the following: very good accuracies can even be reached by correctly estimating a split point over that single feature, and performing classifications using such a single check. Indeed that feature is the only one in which we encoded graph structure, and therefore we expect even better results by a careful design of further similar graph-dependent features.

Including features which depend on dual values, and therefore change at each pricing iteration, proves beneficial when training and testing is performed on different base EVRP instances, thereby indicating that better generalization can be achieved by including them.

Overall, our experiments indicate that integrating simple preprocessing, data driven models and partial inspection heuristics is possible and fruitful.

# 6 Dataset

We did our experiments on three datasets. The full repository is available at https://dataverse.unimi.it/dataverse/optlab in xml format, compliant with the VRP-rep specifications [26]. Each instance was solved to proven optimality. Note that we report in this section the critical resource choice for each instance, but that information is not part of the original data and was found by means of the algorithms described in Section 5.

Size of instances (up to 30 customers) may appear small compared to those solvable in other routing problems; however, the presence of continuous variables and multiple recharge technologies make the problem much more difficult than classical VRP instances of the same size, and we were interested in investigating the mixed-integer structure of the problem rather than in solving large-scale instances.

**Dataset A.** This dataset was derived in [38] from the Solomon dataset, by relaxing the time windows constraints: instances have up to 15 customers (the last part of the name indicates the size of each instance) and 5 stations with a single technology. Some of these instances are very small and not challenging: we solved them mainly to make a comparison between the results of similar problems.

Instances in dataset A are splitted in three classes: C (*clustered*), R (*random*) and RC (*random-clustered*), according to distances between customers. This dataset is summarized in Table 9.

For some instances in this dataset we also modified the number of vehicles with respect to the original value used in [38]. In one case this was done to make the instance feasible, because the original one was not [37]. In some other cases we decreased the number of vehicles to the minimum value for which the instance was known to be feasible [14].

**Dataset B.** Instances in this dataset have 10 customers, up to 5 vehicles, up to 9 stations and 3 technologies. This dataset is summarized in Table 10.

**Dataset C.** This dataset was adapted from the Solomon dataset: all instances have 30 customers, 7 vehicles, 5 stations and 3 technologies. In this dataset customer locations are clustered. This dataset is summarized in Table 11.

**Note on format.** Actual input files include also two constants $\pi$ and $v$. The energy consumption is assumed to be proportional to the distance through coefficient $\pi$, and time consumption is assumed to be proportional to the same distance through coefficient $v$. In other words, being $l_e$ the lenght of an edge $e \in \mathcal{E}$, we can assume $d_e = \pi l_e$ and $t_e = l_e/v$.

## 6.1 Sample instance

Table 8 contains the whole input for instance A-C101-5, the smallest one. This instance has 5 customer vertices and 2 recharge stations other than the depot.

A single recharge technology $h_1$ is available, but we suppose to ignore recharge time for the depot, so we add technology $h_0$ for depot only. Note that the depot can also be used as a "normal" recharge stations with $h_1$; this is modeled with two stations having the same coordinates $[40; 50]$ (see Observation 2.1.2). Figure 9, drawed using the VRP-rep instance mapper [26], also refers to the same instance.

**Optimal solution.** Figure 10a shows the optimal solution for this simple instance. Note that, even if $K = 2$ (that is, we can use two vehicles), the optimal solution consists of a single route visiting every customer. In we enforce the usage of every available vehicle (substituting constraint 3 in the MP with an equality), the best possible solution becomes slightly worse, as shown in Figure 10b.

This is a direct consequence of Observation 2.3.7. In fact, the optimal solution with one route can be obtained from the optimal solution with two routes by a single shortcut (just replace path 7-8-0 with 7-0).

**Observation 6.1.1.** In our formulation of the EVRP problem, time and capacity limits are the only constraints that could stop a single vehicle from visiting the whole graph. If neither time nor capacity are really binding the optimal solution will be always a single route, regardless of graph structure and recharge technologies (as a more general consequence of Observation 2.3.7).

**Table 8:** Details of instance A-C101-5. Note that $Q = 200$ is greater than the overall customer demand $\sum_{i \in \mathcal{N}} q_i = 90$, so we can set $Q = 90$ (see Observation 2.1.1).

| $\|\mathcal{N}\|$ | $\|\mathcal{R}\|$ | $K$ | $Q$ | $T$ | $B$ |
|---|---|---|---|---|---|
| 5 | 3 | 2 | 200* | 1236 | 77 |

CUSTOMERS

| $ID$ | $Xcoord$ | $Ycoord$ | $q$ | $s$ |
|---|---|---|---|---|
| 0 | 20 | 55 | 10 | 90 |
| 1 | 25 | 85 | 20 | 90 |
| 2 | 55 | 85 | 20 | 90 |
| 3 | 68 | 60 | 30 | 90 |
| 4 | 48 | 30 | 10 | 90 |

STATIONS

| $ID$ | $Xcoord$ | $Ycoord$ | $s$ | $Tech$ |
|---|---|---|---|---|
| 5 | 40 | 50 | 0 | $h_1$ |
| 6 | 31 | 84 | 0 | $h_1$ |
| 7 | 39 | 26 | 0 | $h_1$ |
| 8 | 40 | 50 | 0 | $h_0$ |

TECHNOLOGIES

| $Tech$ | $\gamma$ | $\rho$ |
|---|---|---|
| $h_0$ | 1 | 0 |
| $h_1$ | 1 | $3,47$ |

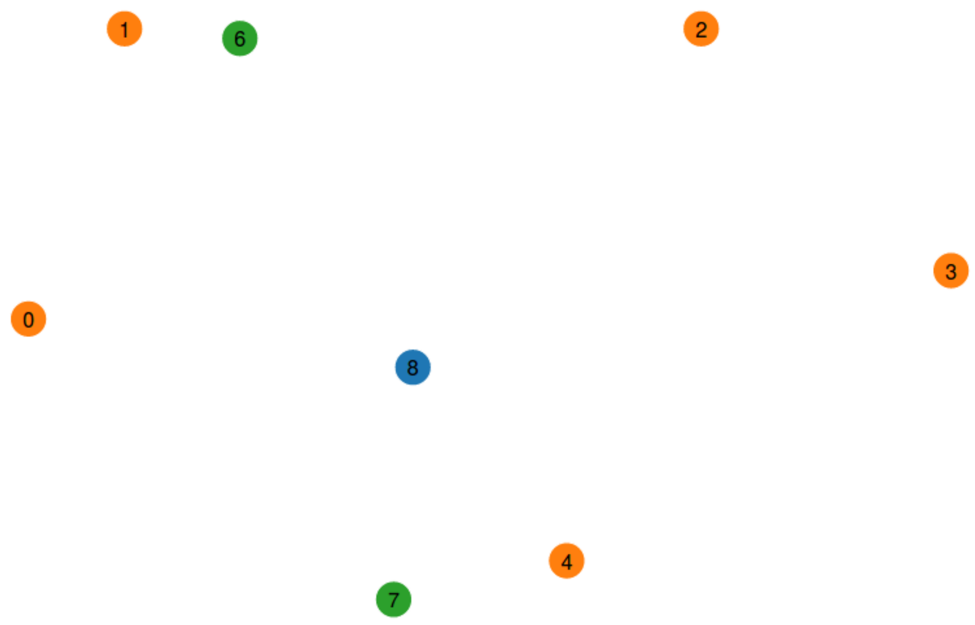
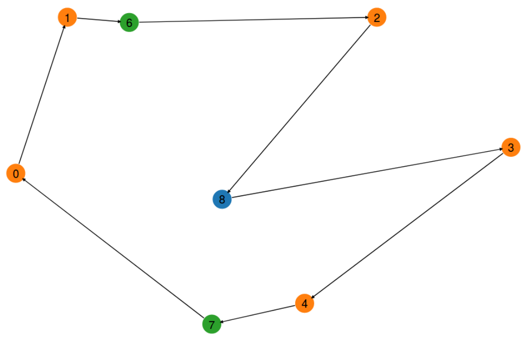
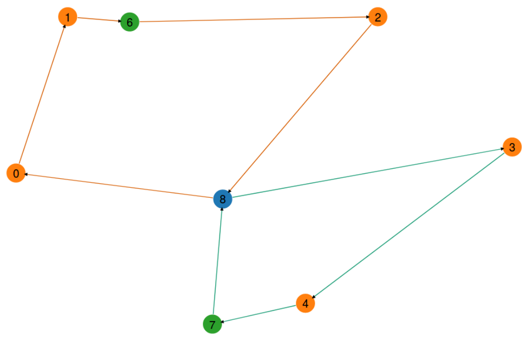


**Figure 9:** Visualization of the graph described in Table 8, with euclidean distances. Station 5 and the depot 8 overlap.



**(a)** Best solution with 1 route (cost: 214)



**(b)** Best solution with 2 routes (cost: 225)

**Figure 10:** Optimal solution for instance A-C101-5.

**Table 9:** Dataset A

| Instance | $\lvert\mathcal{N}\rvert$ | $\lvert\mathcal{R}\rvert$ | $K$ | $Q$ | $T$ | $B$ | c.r. |
|---|---|---|---|---|---|---|---|
| A-C101-10 | 10 | 5 | 2 | 200 | 1236 | 77 | $T$ |
| A-C101-5 | 5 | 3 | 2 | 90 | 1236 | 77 | $Q$ |
| A-C103-15 | 15 | 5 | 2 | 200 | 1236 | 77 | $T$ |
| A-C103-5 | 5 | 2 | 1 | 90 | 1236 | 77 | $Q$ |
| A-C104-10 | 10 | 4 | 2 | 180 | 1236 | 77 | $T$ |
| A-C106-15 | 15 | 3 | 3 | 170 | 1236 | 77 | $T$ |
| A-C202-10 | 10 | 5 | 1 | 220 | 3390 | 77 | $Q$ |
| A-C202-15 | 15 | 5 | 2 | 240 | 3390 | 77 | $Q$ |
| A-C205-10 | 10 | 3 | 2 | 180 | 3390 | 77 | $Q$ |
| A-C206-5 | 5 | 4 | 1 | 70 | 3390 | 77 | $Q$ |
| A-C208-15 | 15 | 4 | 2 | 250 | 3390 | 77 | $Q$ |
| A-C208-5 | 5 | 3 | 1 | 100 | 3390 | 77 | $Q$ |
| A-R102-10 | 10 | 4 | 3 | 155 | 230 | 60 | $T$ |
| A-R102-15 | 15 | 8 | 5 | 191 | 230 | 60 | $T$ |
| A-R103-10 | 10 | 3 | 2 | 139 | 230 | 60 | $T$ |
| A-R104-5 | 5 | 3 | 2 | 86 | 230 | 60 | $T$ |
| A-R105-15 | 15 | 6 | 3 | 200 | 230 | 60 | $T$ |
| A-R105-5 | 5 | 3 | 2 | 58 | 230 | 60 | $T$ |
| A-R201-10 | 10 | 4 | 1 | 181 | 1000 | 60 | $Q$ |
| A-R202-15 | 15 | 6 | 2 | 261 | 1000 | 60 | $Q$ |
| A-R202-5 | 5 | 3 | 1 | 61 | 1000 | 60 | $Q$ |
| A-R203-10 | 10 | 5 | 1 | 109 | 1000 | 60 | $Q$ |
| A-R203-5 | 5 | 4 | 1 | 83 | 1000 | 60 | $Q$ |
| A-R209-15 | 15 | 5 | 1 | 247 | 1000 | 60 | $Q$ |
| A-RC102-10 | 10 | 4 | 3 | 181 | 240 | 77 | $T$ |
| A-RC103-15 | 15 | 5 | 3 | 200 | 240 | 77 | $T$ |
| A-RC105-5 | 5 | 4 | 2 | 145 | 240 | 77 | $T$ |
| A-RC108-10 | 10 | 4 | 3 | 146 | 240 | 77 | $T$ |
| A-RC108-15 | 15 | 5 | 3 | 200 | 240 | 77 | $T$ |
| A-RC108-5 | 5 | 4 | 2 | 109 | 240 | 77 | $T$ |
| A-RC201-10 | 10 | 4 | 1 | 160 | 960 | 77 | $Q$ |
| A-RC202-15 | 15 | 5 | 2 | 233 | 960 | 77 | $Q$ |
| A-RC204-15 | 15 | 7 | 1 | 287 | 960 | 77 | $Q$ |
| A-RC204-5 | 5 | 4 | 1 | 130 | 960 | 77 | $Q$ |
| A-RC205-10 | 10 | 4 | 2 | 154 | 960 | 77 | $Q$ |
| A-RC208-5 | 5 | 3 | 1 | 82 | 960 | 77 | $Q$ |

**Table 10:** Dataset B

| Instance | $\vert\mathcal{N}\vert$ | $\vert\mathcal{R}\vert$ | $K$ | $Q$ | $T$ | $B$ | c.r. |
|---|---|---|---|---|---|---|---|
| B-10-N10 | 10 | 9 | 4 | 230 | 480 | 128 | $T$ |
| B-11-N10 | 10 | 9 | 4 | 230 | 480 | 128 | $T$ |
| B-12-N10 | 10 | 9 | 4 | 230 | 480 | 128 | $T$ |
| B-13-N10 | 10 | 9 | 4 | 230 | 480 | 128 | $T$ |
| B-14-N10 | 10 | 9 | 5 | 230 | 480 | 128 | $T$ |
| B-15-N10 | 10 | 9 | 4 | 230 | 480 | 128 | $T$ |
| B-16-N10 | 10 | 9 | 4 | 230 | 480 | 128 | $T$ |
| B-17-N10 | 10 | 9 | 5 | 230 | 480 | 128 | $T$ |
| B-18-N10 | 10 | 9 | 5 | 230 | 480 | 128 | $T$ |
| B-19-N10 | 10 | 9 | 4 | 230 | 480 | 128 | $T$ |
| B-20-N10 | 10 | 5 | 4 | 230 | 480 | 128 | $T$ |
| B-21-N10 | 10 | 5 | 4 | 230 | 480 | 128 | $T$ |
| B-22-N10 | 10 | 5 | 4 | 230 | 480 | 128 | $T$ |
| B-23-N10 | 10 | 5 | 3 | 230 | 480 | 128 | $T$ |
| B-24-N10 | 10 | 5 | 4 | 230 | 480 | 128 | $T$ |
| B-25-N10 | 10 | 5 | 3 | 230 | 480 | 128 | $T$ |
| B-26-N10 | 10 | 5 | 4 | 230 | 480 | 128 | $T$ |
| B-27-N10 | 10 | 5 | 4 | 230 | 480 | 128 | $T$ |
| B-28-N10 | 10 | 5 | 4 | 230 | 480 | 128 | $T$ |
| B-29-N10 | 10 | 5 | 4 | 230 | 480 | 128 | $T$ |

**Table 11:** Dataset C

| Instance | $\vert\mathcal{N}\vert$ | $\vert\mathcal{R}\vert$ | $K$ | $Q$ | $T$ | $B$ | c.r. |
|---|---|---|---|---|---|---|---|
| C-0-N030 | 30 | 5 | 7 | 2300 | 480 | 160 | $T$ |
| C-1-N030 | 30 | 5 | 6 | 2300 | 480 | 160 | $T$ |
| C-2-N030 | 30 | 5 | 6 | 2300 | 480 | 160 | $T$ |
| C-3-N030 | 30 | 5 | 7 | 2300 | 480 | 160 | $T$ |
| C-4-N030 | 30 | 5 | 7 | 2300 | 480 | 160 | $T$ |
| C-5-N030 | 30 | 5 | 6 | 2300 | 480 | 160 | $T$ |
| C-6-N030 | 30 | 5 | 6 | 2300 | 480 | 160 | $T$ |
| C-7-N030 | 30 | 5 | 6 | 2300 | 480 | 160 | $T$ |
| C-8-N030 | 30 | 5 | 6 | 2300 | 480 | 160 | $T$ |
| C-9-N030 | 30 | 5 | 6 | 2300 | 480 | 160 | $T$ |

# 7 Experimental analysis

We implemented our algorithms in C++, using the SCIP framework [15] version 7.0.1 (including the SoPLEX solver for the LP subproblems).

Let us remark that our algorithm is not intended to be used as a heuristic, and in fact we did not implement a specific heuristic for such a problem, but we rather used an off-the-shelf generic rounding heuristic available in the SCIP framework for the master problem.

The algorithm has been executed on a PC equipped with an 1950X 16-Core processor and 32 GB of RAM, running Linux Ubuntu 18.

## 7.1 Guessing the critical resource

The machine learning approach described in Section 5 can be very useful to fix the critical resource in preprocessing as part of the data, but in order to guess the critical resource at runtime we can adopt a faster (and heuristic) approach. Since we have precomputed a dummy route $r^*$ as described in Subsection 3.2, the critical resource could be inferred from that route only, whose resource consumption values are $\phi_{r^*}$ capacity and $\tau_{r^*}$ time.

We present three different approaches: Algorithm 7 is a simple (and poor) choice of the most consumed resource, Algorithm 8 is a choice based on relative consumption, and Algorithm 9 is the same as Algorithm 8 with a correction based on the available vehicles for time constraint only (every vehicle can consume an arbitrary amount of time up to $T$, while the overall capacity consumption cannot be greater than the sum of customer demands).

Table 12 contains the results of the three strategies, compared with the critical resource label (c.r.) reported in Table 9. Results for dataset B and C are not shown, as for those datasets every strategy always predict T. Accuracy, Precision and Recall metrics are reported in Table 13, from which we can choose Algorithm 9 as the best one.

**Algorithm 7** Absolute resource consumption

```
**if** $T > \tau_{r^*}$ **then return** T
**else**
    **if** $Q > \phi_{r^*}$ **then return** Q
    **elsereturn** T
```

**Algorithm 8** Relative resource consumption

```
**if** $Q > \phi_{r^*}$ **then**
    $\overline{Q} = Q/\phi_{r^*}$
    $\overline{T} = T/\tau_{r^*}$
    **if** $(\overline{Q} > \overline{T})$ **then return** Q
**elsereturn** T
```

**Table 12:** Critical resource of instances in Dataset A

| instance | $\|\mathcal{N}\|$ | $\|\mathcal{R}\|$ | $K$ | $Q$ | $T$ | $B$ | c.r. | $S1$ | $S2$ | $S3$ |
|---|---|---|---|---|---|---|---|---|---|---|
| A-C101-10 | 10 | 5 | 2 | 200 | 1236 | 77 | $T$ | $T$ | $T$ | $T$ |
| A-C101-5 | 5 | 3 | 2 | 90 | 1236 | 77 | $Q$ | $Q$ | $Q$ | $Q$ |
| A-C103-15 | 15 | 5 | 2 | 200 | 1236 | 77 | $T$ | $T$ | $T$ | $T$ |
| A-C103-5 | 5 | 2 | 1 | 90 | 1236 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-C104-10 | 10 | 4 | 2 | 180 | 1236 | 77 | $T$ | $Q$ | $Q$ | $Q$ |
| A-C106-15 | 15 | 3 | 3 | 170 | 1236 | 77 | $T$ | $Q$ | $Q$ | $T$ |
| A-C202-10 | 10 | 5 | 1 | 220 | 3390 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-C202-15 | 15 | 5 | 2 | 240 | 3390 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-C205-10 | 10 | 3 | 2 | 180 | 3390 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-C206-5 | 5 | 4 | 1 | 70 | 3390 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-C208-15 | 15 | 4 | 2 | 250 | 3390 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-C208-5 | 5 | 3 | 1 | 100 | 3390 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-R102-10 | 10 | 4 | 3 | 155 | 230 | 60 | $T$ | $Q$ | $Q$ | $T$ |
| A-R102-15 | 15 | 8 | 5 | 191 | 230 | 60 | $T$ | $Q$ | $Q$ | $T$ |
| A-R103-10 | 10 | 3 | 2 | 139 | 230 | 60 | $T$ | $Q$ | $Q$ | $Q$ |
| A-R104-5 | 5 | 3 | 2 | 86 | 230 | 60 | $T$ | $Q$ | $Q$ | $Q$ |
| A-R105-15 | 15 | 6 | 3 | 200 | 230 | 60 | $T$ | $T$ | $T$ | $T$ |
| A-R105-5 | 5 | 3 | 2 | 58 | 230 | 60 | $T$ | $Q$ | $Q$ | $Q$ |
| A-R201-10 | 10 | 4 | 1 | 181 | 1000 | 60 | $Q$ | $T$ | $Q$ | $Q$ |
| A-R202-15 | 15 | 6 | 2 | 261 | 1000 | 60 | $Q$ | $T$ | $Q$ | $Q$ |
| A-R202-5 | 5 | 3 | 1 | 61 | 1000 | 60 | $Q$ | $T$ | $Q$ | $Q$ |
| A-R203-10 | 10 | 5 | 1 | 109 | 1000 | 60 | $Q$ | $T$ | $Q$ | $Q$ |
| A-R203-5 | 5 | 4 | 1 | 83 | 1000 | 60 | $Q$ | $T$ | $Q$ | $Q$ |
| A-R209-15 | 15 | 5 | 1 | 247 | 1000 | 60 | $Q$ | $Q$ | $Q$ | $Q$ |
| A-RC102-10 | 10 | 4 | 3 | 181 | 240 | 77 | $T$ | $Q$ | $Q$ | $Q$ |
| A-RC103-15 | 15 | 5 | 3 | 200 | 240 | 77 | $T$ | $T$ | $T$ | $T$ |
| A-RC105-5 | 5 | 4 | 2 | 145 | 240 | 77 | $T$ | $Q$ | $Q$ | $T$ |
| A-RC108-10 | 10 | 4 | 3 | 146 | 240 | 77 | $T$ | $Q$ | $Q$ | $Q$ |
| A-RC108-15 | 15 | 5 | 3 | 200 | 240 | 77 | $T$ | $T$ | $T$ | $T$ |
| A-RC108-5 | 5 | 4 | 2 | 109 | 240 | 77 | $T$ | $Q$ | $Q$ | $Q$ |
| A-RC201-10 | 10 | 4 | 1 | 160 | 960 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-RC202-15 | 15 | 5 | 2 | 233 | 960 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-RC204-15 | 15 | 7 | 1 | 287 | 960 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-RC204-5 | 5 | 4 | 1 | 130 | 960 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-RC205-10 | 10 | 4 | 2 | 154 | 960 | 77 | $Q$ | $T$ | $Q$ | $Q$ |
| A-RC208-5 | 5 | 3 | 1 | 82 | 960 | 77 | $Q$ | $T$ | $Q$ | $Q$ |

**Algorithm 9** Relative resource consumption with vehicle correction

```
if $Q > \phi_{r^*}$ then
    $\overline{Q} = Q/\phi_{r^*}$
    $\overline{T} = TK/\tau_{r^*}$                    ▷ K: number of vehicles
    if $(\overline{Q} > \overline{T})$ then return Q
elsereturn T
```

**Table 13:** Comparison between guessing strategies

| Strategy | Accuracy | Precision Q | Precision T | Recall Q | Prec. T |
|---|---|---|---|---|---|
| Algorithm 7 | 0.19 | 0.10 | 0.31 | 0.15 | 0.22 |
| Algorithm 8 | 0.69 | 1.00 | 0.31 | 0.64 | 1.00 |
| Algorithm 9 | 0.81 | 1.00 | 0.56 | 0.74 | 1.00 |

## 7.2 Comparison with path-based approach

Results in this section are compared with results from the path-based approach described in [8], where the pricing algorithm does not compute whole routes but station-to-station paths.

We assume to always select the critical resource according to Algorithm 9.

**Definition 7.2.1.** We define a *path* to be a sequence of customers visited by the same vehicle between two recharge stations (including the depot). We indicate as *empty path* path that directly connects two recharge stations without visiting any customer in between. The two endpoints of a non-empty path can coincide.

The route-based formulation described in Section 3 has a complex mixed-integer pricing sub-problem and a very simple master problem, while the path-based formulation in [8] has a pure integer pricing sub-problem and a much more involved master problem which must handle the ricombination of multiple paths into feasible routes (note that an optimal solution may include empty paths; see Observation 2.3.4). The two formulations, that are equivalent in the discrete domain, do not have equivalent linear relaxations, because the path-based formulation allows for convex combinations of paths that are not routes.

The path-based algorithm implemented in [8] disregards the capacity constraint in the pricing problem, because it is unlikely to be binding in a single path and can be handled directly by the master solver in the join phase. This reduces computation time, but also implies weakening the relaxation of the master problem.

### 7.2.1 Solution quality comparison

In Tables 14, 15 and 16 we report the results obtained while solving the problem to proven optimality with 2 hours timeout. Tables respect the following format:

$|\mathcal{N}|$ Number of customers

$|\mathcal{R}|$ Number of recharge stations

$K$ Number of vehicles available

**Pricer calls** Number of pricer calls

**Nodes** Number of nodes in the branch-and-bound tree

**Time** Execution time in seconds (* indicates a 2 hours timeout)

### 7.2.2 Branch-and-bound comparison

In Tables 17, 18 and 19 we report the primal bound, dual bound and gap obtained while solving the problem to proven optimality with 2 hours timeout. Tables respect the following format:

**PB** Primal bound at the end of branching

**DB** Dual bound at the end of branching (* indicates a 2 hours timeout before closing the root node)

**Gap** Optimality gap

We are grateful to the authors of [8] for their data on primal and dual bounds (they are not actually shown in [8] and were provided directly upon request).

## 7.3 Results

The results show that optimal solution is usually found by our algorithm in the early phases of the branching tree exploration, when only few nodes have been analysed. This is quite intuitive: being tighter, the route-based formulation yields much stronger lower bounds and require fewer branching levels to achieve the integrality of the solutions. Indeed, for the smallest instances the optimal solution was always found at the root node, without branching at all.

Note also that the most time-consuming instances (for our formulation) are not those in dataset B and C, but 15-customers instances in dataset A. This is due to the loose bound on time and capacity for some of those instances. Indeed, the two instances that time out (namely, A-R209-15 and A-RC204-15) have completely ineffective time and capacity constraints, and are indeed ETSP instances (see 3.2.1).

Overall, our formulation (route-based) outperforms the path-based formulation except for instances where both constraints are loose. As a matter of fact, the path-based algorithm practically ignores those contraints anyway (capacity constraint is actually ignored; time constraint is enforced at pricing level but is also unlikely to be binding for a single path).

For some instances in dataset 18 and 19, such as C-3-N030, it could seem inconsistent to have a primal bound (route-based) smaller than a lower bound (path-based). This is due to the path-based formulation's mandatory usage of all vehicles: when the number of routes in the optimal solution is smaller than $K$, using an extra route may worsen the solution quality (and indeed it does).

**Table 14:** Solutions for dataset A

| Instance | $\|\mathcal{N}\|$ | $\|\mathcal{R}\|$ | $K$ | Pricer calls | | B&B nodes | | Time (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | path | route | path | route | path | route |
| A-C101-10 | 10 | 5 | 2 | 36015 | 12 | 71301 | 1 | 4386 | 2.72 |
| A-C101-5 | 5 | 3 | 2 | 320 | 3 | 591 | 1 | 1.72 | 0.03 |
| A-C103-15 | 15 | 5 | 2 | 8092 | 20 | 15875 | 1 | 2871 | 49 |
| A-C103-5 | 5 | 2 | 1 | 43 | 3 | 79 | 1 | 0.09 | 0.03 |
| A-C104-10 | 10 | 4 | 2 | 3665 | 13 | 7201 | 1 | 165 | 3.42 |
| A-C106-15 | 15 | 3 | 3 | 5778 | 77 | 11455 | 3 | 794 | 585 |
| A-C202-10 | 10 | 5 | 1 | 788 | 22 | 1551 | 1 | 53.2 | 63 |
| A-C202-15 | 15 | 5 | 2 | 261 | 38 | 489 | 1 | 67.1 | 657 |
| A-C205-10 | 10 | 3 | 2 | 933 | 13 | 1802 | 1 | 22 | 1.68 |
| A-C206-5 | 5 | 4 | 1 | 5 | 6 | 1 | 1 | 0.04 | 0.10 |
| A-C208-15 | 15 | 4 | 2 | 295 | 56 | 438 | 3 | 29.7 | 6478 |
| A-C208-5 | 5 | 3 | 1 | 73 | 4 | 125 | 1 | 0.28 | 0.03 |
| A-R102-10 | 10 | 4 | 3 | 3672 | 10 | 7207 | 1 | 198 | 0.12 |
| A-R102-15 | 15 | 8 | 5 | 901 | 165 | 862 | 11 | 7200* | 43.6 |
| A-R103-10 | 10 | 3 | 2 | 439 | 9 | 793 | 1 | 10.07 | 8.73 |
| A-R104-5 | 5 | 3 | 2 | 313 | 5 | 611 | 1 | 1.47 | 0.03 |
| A-R105-15 | 15 | 6 | 3 | 15077 | 37 | 15656 | 3 | 7200* | 6.29 |
| A-R105-5 | 5 | 3 | 2 | 975 | 4 | 1928 | 1 | 4.96 | 0.02 |
| A-R201-10 | 10 | 4 | 1 | 100 | 26 | 163 | 1 | 2.12 | 27.0 |
| A-R202-15 | 15 | 6 | 2 | 156 | 31 | 208 | 1 | 62.7 | 142 |
| A-R202-5 | 5 | 3 | 1 | 84 | 2 | 155 | 1 | 0.20 | 0.05 |
| A-R203-10 | 10 | 5 | 1 | 9 | 20 | 1 | 1 | 0.55 | 22.7 |
| A-R203-5 | 5 | 4 | 1 | 34 | 5 | 59 | 1 | 0.19 | 0.04 |
| A-R209-15 | 15 | 5 | 1 | 28 | 64 | 32 | 1 | 3.31 | 7200* |
| A-RC102-10 | 10 | 4 | 3 | 77513 | 16 | 152467 | 3 | 4822 | 1.49 |
| A-RC103-15 | 15 | 5 | 3 | 40870 | 12 | 45417 | 1 | 7200* | 0.37 |
| A-RC105-5 | 5 | 4 | 2 | 6444 | 5 | 12843 | 1 | 65.6 | 0.03 |
| A-RC108-10 | 10 | 4 | 3 | 33827 | 6 | 66936 | 1 | 2075 | 0.36 |
| A-RC108-15 | 15 | 5 | 3 | 34319 | 32 | 43382 | 3 | 7200* | 1.77 |
| A-RC108-5 | 5 | 4 | 2 | 2647 | 3 | 5031 | 1 | 18.1 | 0.02 |
| A-RC201-10 | 10 | 4 | 1 | 284 | 26 | 547 | 1 | 9.82 | 21.2 |
| A-RC202-15 | 15 | 5 | 2 | 5017 | 41 | 9499 | 1 | 1336 | 326 |
| A-RC204-15 | 15 | 7 | 1 | 1237 | 74 | 2305 | 1 | 863 | 7200* |
| A-RC204-5 | 5 | 4 | 1 | 209 | 7 | 401 | 1 | 1.07 | 0.07 |
| A-RC205-10 | 10 | 4 | 2 | 10146 | 11 | 20177 | 1 | 443 | 0.60 |
| A-RC208-5 | 5 | 3 | 1 | 118 | 3 | 212 | 1 | 0.43 | 0.03 |

**Table 15:** Solutions for dataset B

| Instance | $\|\mathcal{N}\|$ | $\|\mathcal{R}\|$ | $K$ | Pricer calls | | B&B nodes | | Time (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | path | route | path | route | path | route |
| B-10-N10 | 10 | 9 | 4 | 560 | 3 | 635 | 3 | 4740 | 0.43 |
| B-11-N10 | 10 | 9 | 4 | 280 | 6 | 366 | 1 | 3664 | 0.06 |
| B-12-N10 | 10 | 9 | 4 | 813 | 6 | 814 | 1 | 7200* | 0.11 |
| B-13-N10 | 10 | 9 | 4 | 523 | 7 | 526 | 1 | 7200* | 0.14 |
| B-14-N10 | 10 | 9 | 5 | 856 | 17 | 826 | 3 | 7200* | 0.20 |
| B-15-N10 | 10 | 9 | 4 | 110 | 17 | 128 | 3 | 1244 | 0.40 |
| B-16-N10 | 10 | 9 | 4 | 379 | 59 | 433 | 11 | 7200* | 2.03 |
| B-17-N10 | 10 | 9 | 5 | 373 | 18 | 389 | 3 | 4174 | 0.17 |
| B-18-N10 | 10 | 9 | 5 | 336 | 16 | 414 | 1 | 4544 | 0.16 |
| B-19-N10 | 10 | 9 | 4 | 185 | 29 | 255 | 7 | 3888 | 1.16 |
| B-20-N10 | 10 | 5 | 4 | 167 | 12 | 172 | 3 | 101 | 0.54 |
| B-21-N10 | 10 | 5 | 4 | 279 | 18 | 399 | 2 | 379 | 1.16 |
| B-22-N10 | 10 | 5 | 4 | 1739 | 24 | 3116 | 5 | 2614 | 0.64 |
| B-23-N10 | 10 | 5 | 3 | 3660 | 41 | 3694 | 11 | 7200* | 0.56 |
| B-24-N10 | 10 | 5 | 4 | 5223 | 13 | 6396 | 5 | 7200* | 0.83 |
| B-25-N10 | 10 | 5 | 3 | 3280 | 7 | 3260 | 3 | 7200* | 0.22 |
| B-26-N10 | 10 | 5 | 4 | 5190 | 14 | 8993 | 3 | 7200* | 0.48 |
| B-27-N10 | 10 | 5 | 4 | 2118 | 39 | 3675 | 7 | 1760 | 0.56 |
| B-28-N10 | 10 | 5 | 4 | 185 | 12 | 309 | 3 | 110 | 0.23 |
| B-29-N10 | 10 | 5 | 4 | 1579 | 14 | 2141 | 3 | 744 | 0.74 |

**Table 16:** Solutions for dataset C

| Instance | $\|\mathcal{N}\|$ | $\|\mathcal{R}\|$ | $K$ | Pricer calls | | B&B nodes | | Time (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | path | route | path | route | path | route |
| C-0-N030 | 30 | 5 | 7 | 57 | 82 | 84 | 5 | 849 | 7.34 |
| C-1-N030 | 30 | 5 | 6 | 2334 | 245 | 2350 | 13 | 7200* | 20.0 |
| C-2-N030 | 30 | 5 | 6 | 1429 | 106 | 1426 | 7 | 7200* | 11.9 |
| C-3-N030 | 30 | 5 | 7 | 1883 | 87 | 2152 | 3 | 6910 | 3.33 |
| C-4-N030 | 30 | 5 | 7 | 1475 | 156 | 1631 | 9 | 7200* | 9.97 |
| C-5-N030 | 30 | 5 | 6 | 14 | 190 | 4 | 7 | 7200* | 37.1 |
| C-6-N030 | 30 | 5 | 6 | 1218 | 310 | 1307 | 15 | 4611 | 28.7 |
| C-7-N030 | 30 | 5 | 6 | 6 | 25 | 1 | 1 | 247 | 0.87 |
| C-8-N030 | 30 | 5 | 6 | 167 | 99 | 170 | 5 | 1125 | 4.86 |
| C-9-N030 | 30 | 5 | 6 | 61 | 17 | 63 | 1 | 951 | 0.61 |

**Table 17:** Full B&B results for dataset A

| Instance | PB | | DB | | Gap | |
|---|---|---|---|---|---|---|
| | path | route | path | route | path | route |
| A-C101-10 | 303.00 | 303.00 | 302.70 | 303.00 | 0.10% | 0.00% |
| A-C101-5 | 214.00 | 214.00 | 214.00 | 214.00 | 0.00% | 0.00% |
| A-C103-15 | 290.00 | 290.00 | 289.71 | 290.00 | 0.10% | 0.00% |
| A-C103-5 | 157.00 | 157.00 | 157.00 | 157.00 | 0.00% | 0.00% |
| A-C104-10 | 281.00 | 281.00 | 280.72 | 281.00 | 0.10% | 0.00% |
| A-C106-15 | 253.00 | 253.00 | 252.75 | 253.00 | 0.10% | 0.00% |
| A-C202-10 | 234.00 | 234.00 | 233.79 | 234.00 | 0.09% | 0.00% |
| A-C202-15 | 332.00 | 332.00 | 331.67 | 332.00 | 0.10% | 0.00% |
| A-C205-10 | 233.00 | 233.00 | 232.83 | 233.00 | 0.07% | 0.00% |
| A-C206-5 | 205.00 | 205.00 | 205.00 | 205.00 | 0.00% | 0.00% |
| A-C208-15 | 269.00 | 269.00 | 268.75 | 269.00 | 0.09% | 0.00% |
| A-C208-5 | 161.00 | 161.00 | 161.00 | 161.00 | 0.00% | 0.00% |
| A-R102-10 | 230.00 | 230.00 | 229.81 | 230.00 | 0.08% | 0.00% |
| A-R102-15 | 345.00 | 317.00 | 269.22 | 317.00 | 28.15% | 0.00% |
| A-R103-10 | 171.00 | 169.00 | 168.17 | 169.00 | 1.68% | 0.00% |
| A-R104-5 | 142.00 | 142.00 | 142.00 | 142.00 | 0.00% | 0.00% |
| A-R105-15 | 308.00 | 297.00 | 238.67 | 297.00 | 29.05% | 0.00% |
| A-R105-5 | 160.00 | 160.00 | 159.88 | 160.00 | 0.08% | 0.00% |
| A-R201-10 | 189.00 | 189.00 | 188.92 | 189.00 | 0.04% | 0.00% |
| A-R202-15 | 286.00 | 286.00 | 285.86 | 286.00 | 0.05% | 0.00% |
| A-R202-5 | 147.00 | 147.00 | 147.00 | 147.00 | 0.00% | 0.00% |
| A-R203-10 | 252.00 | 252.00 | 252.00 | 252.00 | 0.00% | 0.00% |
| A-R203-5 | 185.00 | 185.00 | 185.00 | 185.00 | 0.00% | 0.00% |
| A-R209-15 | 264.00 | 376.00 | 264.00 | $*$ | 0.00% | $*$ |
| A-RC102-10 | 428.00 | 428.00 | 427.57 | 428.00 | 0.10% | 0.00% |
| A-RC103-15 | 387.00 | 367.00 | 296.67 | 367.00 | 30.45% | 0.00% |
| A-RC105-5 | 220.00 | 220.00 | 219.80 | 220.00 | 0.09% | 0.00% |
| A-RC108-10 | 355.00 | 355.00 | 354.65 | 355.00 | 0.10% | 0.00% |
| A-RC108-15 | 384.00 | 384.00 | 358.50 | 384.00 | 7.11% | 0.00% |
| A-RC108-5 | 259.00 | 259.00 | 258.74 | 259.00 | 0.10% | 0.00% |
| A-RC201-10 | 258.00 | 258.00 | 257.79 | 258.00 | 0.08% | 0.00% |
| A-RC202-15 | 305.00 | 305.00 | 304.70 | 305.00 | 0.10% | 0.00% |
| A-RC204-15 | 295.00 | 484.00 | 294.72 | $*$ | 0.09% | $*$ |
| A-RC204-5 | 182.00 | 182.00 | 182.00 | 182.00 | 0.00% | 0.00% |
| A-RC205-10 | 320.00 | 320.00 | 319.71 | 320.00 | 0.09% | 0.00% |
| A-RC208-5 | 172.00 | 172.00 | 172.00 | 172.00 | 0.00% | 0.00% |

**Table 18:** Full B&B results for dataset B

| Instance | PB | | DB | | Gap | |
|---|---|---|---|---|---|---|
| | path | route | path | route | path | route |
| B-10-N10 | 22.92 | 22.92 | 22.91 | 22.92 | 0.06% | 0.00% |
| B-11-N10 | 24.86 | 24.86 | 24.83 | 24.86 | 0.10% | 0.00% |
| B-12-N10 | 24.15 | 23.51 | 22.93 | 23.51 | 5.31% | 0.00% |
| B-13-N10 | 25.04 | 23.70 | 21.99 | 23.70 | 13.85% | 0.00% |
| B-14-N10 | ∗ | 31.26 | 28.98 | 31.26 | ∗ | 0.00% |
| B-15-N10 | 21.13 | 20.24 | 21.11 | 20.24 | 0.08% | 0.00% |
| B-16-N10 | 20.20 | 20.20 | 19.73 | 20.20 | 2.38% | 0.00% |
| B-17-N10 | 28.88 | 28.11 | 28.86 | 28.11 | 0.07% | 0.00% |
| B-18-N10 | 28.07 | 25.39 | 28.05 | 25.39 | 0.07% | 0.00% |
| B-19-N10 | 20.11 | 20.11 | 20.11 | 20.11 | 0.04% | 0.00% |
| B-20-N10 | 19.24 | 19.24 | 19.22 | 19.24 | 0.08% | 0.00% |
| B-21-N10 | 20.29 | 19.99 | 20.27 | 19.99 | 0.10% | 0.00% |
| B-22-N10 | 20.17 | 20.17 | 20.15 | 20.17 | 0.10% | 0.00% |
| B-23-N10 | ∗ | 23.21 | 20.86 | 23.21 | ∗ | 0.00% |
| B-24-N10 | 22.60 | 22.46 | 21.33 | 22.46 | 5.94% | 0.00% |
| B-25-N10 | ∗ | 23.26 | 20.59 | 23.26 | ∗ | 0.00% |
| B-26-N10 | 20.98 | 20.98 | 20.85 | 20.98 | 0.64% | 0.00% |
| B-27-N10 | 24.49 | 24.49 | 24.47 | 24.49 | 0.10% | 0.00% |
| B-28-N10 | 22.51 | 22.51 | 22.49 | 22.51 | 0.09% | 0.00% |
| B-29-N10 | 22.26 | 22.26 | 22.24 | 22.26 | 0.09% | 0.00% |

**Table 19:** Full B&B results for dataset C

| Instance | PB | | DB | | Gap | |
|---|---|---|---|---|---|---|
| | path | route | path | route | path | route |
| C-0-N030 | 39.01 | 39.01 | 38.99 | 39.01 | 0.04% | 0.00% |
| C-1-N030 | 1918.43 | 39.78 | 37.66 | 39.78 | 4993% | 0.00% |
| C-2-N030 | ∗ | 36.76 | 36.43 | 36.76 | ∗ | 0.00% |
| C-3-N030 | 41.62 | 40.44 | 41.58 | 40.44 | 0.10% | 0.00% |
| C-4-N030 | 43.08 | 42.72 | 41.70 | 42.72 | 3.30% | 0.00% |
| C-5-N030 | ∗ | 31.80 | 31.48 | 31.80 | ∗ | 0.00% |
| C-6-N030 | 37.62 | 37.62 | 37.59 | 37.62 | 0.10% | 0.00% |
| C-7-N030 | 30.64 | 30.64 | 30.64 | 30.64 | 0.00% | 0.00% |
| C-8-N030 | 33.83 | 33.83 | 33.83 | 33.83 | 0.01% | 0.00% |
| C-9-N030 | 33.53 | 33.52 | 33.51 | 33.52 | 0.05% | 0.00% |

# 8 Conclusions

In this thesis we propose an exact, effective branch-and-price optimization algorithm for the EVRP problem, employing bi-directional labeling algorithms for generating feasible routes. The problem we have addressed proved to be definitely challenging: the presence of continuous variables and the resulting mixed-integer master problem make the problem much more difficult than classical VRP instances of the same size. Our pricing algorithm yields strong primal solutions, and as a consequence only few branch-and-bound nodes need to be explored for every instance.

We perform a theoretical investigation and exploited properties on the structure of the solution to obtain an exact algorithm for such a problem. We also identify classes of instances that are computationally harder to solve.

Finally, we present a full dataset of instances solved to proven optimality.

## 8.1 Further research

Many extensions are possible both to make the model more realistic and possibly to develop better algorithms. On the model side, besides incorporating constraints that also appear in classical VRP variants (such as time windows, heterogeneous fleets, pick-up and delivery, loading constraints etc.), it is worth to mention some extensions that are specific of the Electric VRP variants:

- Synchronization constraints on simultaneous use of recharge stations by different vehicles;
- Presence of edges with replenishment corresponding to downhill roads (where energy is accumulated instead of being consumed);
- Non-constant energy consumption along edges (see Observation 2.1.5);
- Time-dependent price of energy;
- Capacitated recharging stations;
- Possibility of recharging even while visiting customer vertices.

On the side of possible improvements to the algorithms, we mention the development of a parallel implementation of the join procedure described in Subsection 4.4 for the pricing algorithm: once the bi-directional extension phase has been performed, each route can be rebuilt independently.

As another possible improvement, cutting planes can be used to strengthen the linear relaxations in the branch-and-bound search tree (for example, 2-path cuts, introduced in [22] for the VRPTW). They were not included in the current algorithm since the number of nodes in the branch-and-bound tree appears to be very small for all instances anyway.

We conclude by sketching a possible extension of our branch-and-price approach to other VRP variants, such as *drone routing*. This EVRP variant has aerial vehicles (drones) instead of land vehicles. In general, drones are less expensive to maintain than traditional delivery vehicles and are not limited by established infrastructure such as roads, but are subject to kinematic constraints preventing them to make on-the-spot turns. In this scenario, the model described in Subsection 2.1 remains almost the same, with the only addition of the nonlinear constraint on minimum turn radius. This does not affect the master problem or the branching rules but the the pricing algorithm only.

## Bibliography

[1] Anagnostopoulou Afroditi et al. “Electric vehicle routing problem with industry constraints: Trends and insights for future research”. In: *Transportation Research Procedia* 3 (2014), pp. 452–459 (cit. on p. 4).

[2] Juho Andelmin and Enrico Bartolini. “An Exact Algorithm for the Green Vehicle Routing Problem”. In: *Transportation Science* 51.4 (3027), pp. 1288–1303 (cit. on p. 4).

[3] Dario Bezzi, Alberto Ceselli, and Giovanni Righini. “Automated Tuning of a Column Generation Algorithm”. In: *International Conference on Learning and Intelligent Optimization*. 2020, pp. 201–215 (cit. on p. 6).

[4] Claudia Bongiovanni, Mor Kaspi, and Nikolas Geroliminis. “The electric autonomous dial-a-ride problem”. In: *Transportation Research Part B: Methodological* 122 (2019), pp. 436–456 (cit. on p. 5).

[5] Maurizio Bruglieri, Simona Mancini, and Ornella Pisacane. “The green vehicle routing problem with capacitated alternative fuel stations”. In: *Computers & Operations Research* 112 (2019), p. 104759 (cit. on p. 4).

[6] Maurizio Bruglieri et al. “A variable neighborhood search branching for the electric vehicle routing problem with time windows”. In: *Electronic Notes Discrete Math* 47 (2015), pp. 221–228 (cit. on p. 4).

[7] Maurizio Bruglieri et al. “A path-based solution approach for the Green Vehicle Routing Problem”. In: *Computers and Operations Research* 103 (2019), pp. 109–122 (cit. on p. 4).

[8] Alberto Ceselli et al. “A Branch-and-Cut-and-Price Algorithm for the Electric Vehicle Routing Problem with Multiple Technologies”. In: *SN Operations Research Forum*. Vol. 2. 1. Springer. 2021, pp. 1–33 (cit. on pp. 5, 6, 51, 52).

[9] Ryan David and Brian A. Foster. “An Integer Programming Approach to Scheduling”. In: *Computer scheduling of public transport: Urban passenger vehicle and crew scheduling* (1981), pp. 269–280 (cit. on p. 31).

[10] Guy Desaulniers et al. “Exact Algorithms for Electric Vehicle-Routing Problems with Time Windows”. In: *Operations Research* 64.6 (2016), pp. 1388–1405 (cit. on pp. 4, 14).

[11] Moshe Dror and Pierre Trudeau. “Split delivery routing”. In: *Naval Research Logistics* 37 (1990), pp. 383–402 (cit. on pp. 17, 20).

[12] Sevgi Erdoğan and Elise Miller-Hooks. “A Green Vehicle Routing Problem”. In: *Transportation Research* Part E 48 (2012) (cit. on p. 3).

[13] Dominique Feillet et al. “An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems”. In: *Networks* 44.3 (2004), pp. 216–229 (cit. on p. 24).

[14] Àngel Felipe et al. “A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges”. In: *Transportation Research* Part E 71 (2014) (cit. on pp. 5, 7, 44).

[15] Gerald Gamrath et al. *The SCIP Optimization Suite 7.0.* Technical Report. Optimization Online, 2020. URL: http://www.optimization-online.org/DB_HTML/2020/03/7705.html (cit. on pp. 37, 49).

[16] Dominik Goeke and Michael Schneider. “Routing a mixed fleet of electric and conventional vehicles”. In: *European Journal of Operational Research* 245.1 (2015), pp. 81–99 (cit. on p. 8).

[17] Stefan Irnich and Guy Desaulniers. “Shortest path problems with resource constraints”. In: *Column generation.* Springer, 2005, pp. 33–65 (cit. on p. 24).

[18] Han Jiawei, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques.* Third Edition. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2012 (cit. on p. 37).

[19] Merve Keskin, Gilbert Laporte, and Bülent Çatay. “Electric vehicle routing problem with time-dependent waiting times at recharging stations”. In: *Computers & Operations Research* 107 (2019), pp. 77–94 (cit. on p. 3).

[20] Merve Keskin and Bülent Çatay. “Partial recharge strategies for the electric vehicle routing problem with time windows”. In: *Transportation Research Part C: Emerging Technologies* 65 (2016), pp. 111–127 (cit. on p. 4).

[21] Çağrı Koç, Ola Jabali, and Gilbert Laporte. “Long-haul vehicle routing and scheduling with idling options”. In: *Journal of the operational research society* (2017), pp. 1–13 (cit. on p. 5).

[22] Niklas Kohl et al. “2-path cuts for the vehicle routing problem with time windows”. In: *Transportation Science* 33.1 (1999), pp. 101–116 (cit. on p. 57).

[23] Çağrı Koç and Ismail Karaoglan. “The green vehicle routing problem: A heuristic based exact solution approach”. In: *Applied Soft Computing* 39 (2016), pp. 154 –164 (cit. on p. 3).

[24] Valeria Leggieri and Mohamed Haouari. “A practical solution approach for the green vehicle routing problem”. In: *Transportation Research Part E: Logistics and Transportation Review* 104 (Aug. 2017), pp. 97–112 (cit. on p. 4).

[25] Wang Li-ying and Song Yuan-bin. “Multiple Charging Station Location-Routing Problem with Time Window of Electric Vehicle.” In: *Journal of Engineering Science & Technology Review* 8.5 (2015) (cit. on p. 5).

[26] Jorge E. Mendoza et al. *VRP-REP: the vehicle routing community repository.* 2014. URL: https://vrp-rep.github.io/mapper/ (cit. on pp. 44, 45).

[27] Ming Meng and Yun Ma. “Route Optimization of Electric Vehicle considering Soft Time Windows and Two Ways of Power Replenishment”. In: *Advances in Operations Research* 2020 (2020), pp. 436–456 (cit. on p. 4).

[28] David Meyer et al. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien.* R package version 1.7-2 — For new features, see the Changelog file (in the package source). 2019. URL: https://CRAN.R-project.org/package=e1071 (cit. on p. 38).

[29] Alejandro Montoya et al. “A modified multi space sampling heuristic for the green vehicle routing problem”. In: *Transportation Research* Part C (2015) (cit. on p. 3).

[30] Alejandro Montoya et al. “The electric vehicle routing problem with nonlinear charging function”. In: *Transportation Research Part B: Methodological* 103 (2017), pp. 87–110 (cit. on p. 4).

[31] Ash Omidvar and Reza Tavakkoli-Moghaddam. “Sustainable vehicle routing: Strategies for congestion management and refueling scheduling”. In: *2012 IEEE international energy conference and exhibition (ENERGYCON).* IEEE. 2012, pp. 1089–1094 (cit. on p. 3).

[32] Samuel Pelletier, Ola Jabali, and Gilbert Laporte. “Goods Distribution with Electric Vehicles: Review and Research Perspectives”. In: *Transportation Science* 50 (2015), pp. 3–22 (cit. on p. 3).

[33] Henning Preis, Stefan Frank, and Karl Nachtigall. “Energy-optimized routing of electric vehicles in urban delivery systems”. In: *Operations Research Proceedings 2012.* Springer, 2014, pp. 583–588 (cit. on p. 4).

[34] Giovanni Righini and Matteo Salani. “Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints”. In: *Discrete Optimization* 3.3 (2006), pp. 255 –273 (cit. on p. 25).

[35] Ons Sassi, Wahiba Ramdane Cherif, and Ammar Oulamara. “Vehicle routing problem with mixed fleet of conventional and heterogenous electric vehicles and time dependent charging costs”. In: (2014). URL: https://hal.archives-ouvertes.fr/hal-01083966/document (cit. on p. 5).

[36] Maximilian Schiffer and Grit Walther. “The electric location routing problem with time windows and partial recharging”. In: *European Journal of Operational Research* 260.3 (2017), pp. 995–1013 (cit. on p. 4).

[37] Michael Schneider. *Personal communication.* 2014 (cit. on p. 44).

[38] Michael Schneider, Andreas Stenger, and Dominik Goeke. “The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations”. In: *Transportation Science* 48 (2014), pp. 500–520 (cit. on pp. 3, 4, 44).

[39] Timothy M. Sweda, Irina S. Dolinskaya, and Diego Klabjan. “Optimal Recharging for Electric Vehicles”. In: *Transportation Science* 51.2 (2017), pp. 457–479 (cit. on p. 4).

[40] Terry Therneau and Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15 — For new features, see the Changelog file (in the package source). 2019. URL: `https://CRAN.R-project.org/package=rpart` (cit. on p. 38).

[41] Chen Tianqi, Tong He, and Michael Benesty. *xgboost: Extreme Gradient Boosting*. R package version 0.4-2 — For new features, see the Changelog file (in the package source). 2019. URL: `https://CRAN.R-project.org/package=xgboost` (cit. on p. 38).

[42] Christian Tilk et al. “Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster”. In: *European Journal of Operational Research* 261.2 (2017), pp. 530–539 (cit. on p. 25).