SN Oper. Res. Forum (2021) 2: 8
https://doi.org/10.1007/s43069-020-00052-x

ORIGINAL RESEARCH

A Branch-and-Cut-and-Price Algorithm for the Electric Vehicle Routing Problem with Multiple Technologies

Alberto Ceselli[1] · Ángel Felipe[2] · M. Teresa Ortuño[2] · Giovanni Righini[1] · Gregorio Tirado[3]

Received: 9 March 2020 / Accepted: 22 December 2020 / Published online: 26 January 2021

Abstract
We provide an exact optimization algorithm for the electric vehicle routing problem with multiple recharge technologies. Our branch-and-cut-and-price algorithm relies upon a path-based formulation, where each column in the master problem represents a sequence of customer visits between two recharge stations instead of a whole route. This allows for massive decomposition, and parallel implementation of the pricing phase, exploiting the large number of independent pricing sub-problems. The algorithm could solve instances with up to thirty customers, nine recharge stations, five vehicles and three technologies to proven optimality. Near-optimal heuristic solutions were obtained with a general-purpose MIP solver from the columns generated at the root node.

Keywords Electric vehicle routing · Column generation · Cutting planes · Dynamic programming

1 Introduction

The electric vehicle routing problem (EVRP) is a variation of the vehicle routing problem (VRP) in which the fleet is made of electric vehicles (EVs). The use of EVs in distribution logistics has been extensively investigated by Pelletier et al. [22]. New research on optimization techniques is required because of the specific characteristic of EVs, especially their limited autonomy: EVs may need to recharge their batteries to be able to serve all customers assigned to their routes. The recharge can be achieved by visiting suitable recharge stations, available at known sites in the road network. Section 1.

This article is part of the Topical Collection on *Decomposition at 70*

Alberto Ceselli
alberto.ceselli@unimi.it

Extended author information available on the last page of the article

The EVRP 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.

The scientific literature on the EVRP has rapidly developed, starting from the seminal papers by Conrad and Figliozzi [9] and Erdogan and Miller-Hooks [11]. An up-to-date and comprehensive survey can be found in Keskin, Laporte and Catay [16], where the authors list 49 EVRP papers, classifying them according to the assumptions on the recharge policy, the consumption function, the fleet composition, the presence of multiple recharge technologies, the objective function terms and the constraints such as time windows and capacities.

In this paper, we concentrate on the EVRP with multiple recharge technologies, first introduced by Felipe et al. [13], where each recharge station may be equipped with one or more recharge technologies and each technology is characterized by a different recharge rate and energy price, so that faster recharges are possible but they are more expensive. The need of selecting the optimal technology for each recharge operation adds significant complexity to the problem. In presence of time constraints, such as constraints on the maximum duration of routes, it may be necessary to select a faster recharge technology even if it is more expensive; on the contrary, using a cheaper technology may allow for cost reduction when time is not a binding resource.

The relevance of this variation of the EVRP has been shown in the literature. In their survey Keskin, Laporte and Catay [16] mention several papers dealing with the EVRP with multiple technologies and all of them proposed heuristic algorithms: Sassi, Cherif and Oulamara [23], Li-Ying and Yuan-Bin [19], Sweda, Dolinskaya and Klabjan [27] (single-vehicle), Montoya et al. [21], Keskin and Catay [15], Villegas et al. [28] and Koc, Jabali and Laporte [17].

In this paper, we propose an exact optimization branch-and-cut-and-price algorithm for the EVRP with multiple technologies. In addition, we also consider a cost term due to battery amortization, represented by a fixed cost for each recharge operation, as well as constraints on maximum route duration, maximum number of available vehicles and vehicle capacity.

The BCP algorithm we have developed relies upon a path-based formulation, where each column in the master problem represents a sequence of customer visits in between two recharge stations instead of a whole route; such an unusual approach allows us to get insights into the properties of such an interesting combinatorial substructure. Indeed, similar philosophies have been exploited in the context of airline transportation since seminal papers like [2]: sequences of connected flights that begin and end at maintenance stations, usually termed strings, play the role of our paths. Furthermore, our formulation allows massive decomposition and is therefore amenable to be solved with a parallel column generation algorithm. Our computational results show that it provides an appealing trade-off between the quality of the bounds and the computing effort. In particular, our algorithms prove to scale very well in the number of CPU cores available for computations.

Our computational tests have been done on benchmark instances from the VRP literature, suitably modified to include multiple technologies. The BCP algorithm could solve instances with up to thirty customers, nine recharge stations, five vehicles and three technologies to proven optimality. We also evaluated the quality of the heuristic solutions that can be obtained from the BCP algorithm with rounding techniques, benchmarking them against tailored local search algorithms from the literature. The results show that the presence of multiple technologies makes the EVRP considerably harder and that the performances of the BCP algorithm heavily depend on the structure of the instances (e.g., clustered vs non-clustered customers) and not only on their size.

2 Problem Description

Let $\mathcal{G} = (\mathcal{N} \cup \mathcal{R}, \mathcal{E})$ be a given weighted undirected graph whose vertex set is the union of a set $\mathcal{N}$ of N customers and a set $\mathcal{R}$ of R recharge stations. A distinguished station in $\mathcal{R}$ is the depot, numbered 0, where vehicle routes start and terminate.

All customer vertices in $\mathcal{N}$ must be visited by a single vehicle; split delivery is not allowed. Each customer $i \in \mathcal{N}$ is characterized by a demand q_i.

Stations, i.e., vertices in $\mathcal{R}$, can be visited at any time if needed. Multiple visits to them (also simultaneously) and partial recharges are also allowed. We consider a set $\mathcal{H}$ of different technologies for battery recharge. For each technology $h \in \mathcal{H}$ we assume a given recharge speed ρ_h and a given recharge unit cost γ_h. At each visit to a recharge station, vehicles are allowed to use only one of the technologies available at the station. We denote as $\mathcal{H}_j \subseteq \mathcal{H}$ the set of technologies available at each station $j \in \mathcal{R}$.

All vertices $i \in \mathcal{R} \cup \mathcal{N}$ are also characterized by a service time s_i. In the case of customers, it represents the time taken by delivery operations; in the case of recharge stations, it represents a fixed time to be spent to set up the recharge operations, independently of the amount of recharge. It does not include the actual recharging time.

Non-negative coefficients t_a and e_a are associated with each edge $a \in \mathcal{E}$, to, respectively, represent the time and the energy consumption for traveling along a in either direction.

We consider a fleet made of a set $\mathcal{K}$ of K identical vehicles with given capacity Q and equipped with batteries of given capacity B. The duration of each route is required to be within a given limit T representing the duration of drivers' work shifts.

A feasible route is a closed walk complying with the following set of constraints:

- the route must include the depot;
- the sum of the demands of the customers visited along the route must not exceed the vehicle capacity;
- the total duration of the route must not exceed the total allowed duration T; the route duration is given by three terms:
  - traveling time, i.e., the sum of the terms t_a for each edge $a \in \mathcal{E}$ in the route;
  - service time, i.e., the sum of the terms s_i for each vertex $i \in \mathcal{N} \cup \mathcal{R}$ visited along the route;

  - recharge time at the stations (which is a decision variable, owing to the possibility of partial recharges), excluding the depot;
- the level of battery charge must be kept between 0 and B at any time, taking into account that:
  - the amount of energy consumed by a vehicle along a path is the sum of the terms e_a for each edge $a \in \mathcal{E}$ in the path;
  - the amount of energy recharged at any station $i \in \mathcal{R}$ is given by ρ_i times the (variable) recharge time at $i \in \mathcal{R}$.

A set of feasible routes is a feasible solution if all customers are visited once and no more than K vehicles are used.

As opposed to classical vehicle routing problems, where one wants to minimize the overall distance traveled, the objective to be optimized is the overall recharge cost, consisting of a fixed cost and a variable cost. Since batteries allow for a limited number of recharge cycles during their operational life, we associate a fixed cost with each recharge operation; this cost, indicated by f, is given by the cost of a battery divided by the estimated number of recharge cycles after which the battery must be replaced. The variable cost represents the usual objective function depending on the total distance traveled, but it also depends on the recharge technology selected at each visited station. At any station $i \in R$ the variable cost associated with a recharge operation is proportional to the amount of recharged energy, but it also depends on the chosen recharge technology $h \in \mathcal{H}_i$ through a coefficient γ_h.

3 The Model

Branch-and-cut-and-price (BCP) algorithms have been proposed to solve different variations of the EVRP. Recent contributions include Desaulniers et al. [10], Hiermann et al. [14], Andelmin and Bartolini [1], Breunig et al. [6], Bruglieri, Mancini and Pisacane [7]. All these papers consider the problem with single technology and sometimes with even more simplistic assumptions, as in the cases with no partial recharge allowed and with fixed recharge time. In these BCP algorithms, each column corresponds to a feasible route, which is common in the VRP literature.

In this paper, we investigate a different formulation, that exploits the particular structure of the EVRP. In our model each column corresponds to a feasible path, i.e., a sequence of vertices visited between two recharge stations without any other recharge station in between. This choice is motivated by two main observations.

First, each feasible route can be decomposed into feasible paths and feasible paths can be computed independently for each pair of recharge stations. Therefore, the pricing sub-problem can be solved in parallel and independently for each pair of recharge stations. As shown in Section 5, our computational results show that the features of the path-based formulation nicely fit the advantages offered by a parallel implementation.

Second, customers must be visited only once while recharge stations can be visited multiple times, also by the same vehicle. A possible approach is therefore to develop models with node duplication in which several copies of each station are included in the graph, to correctly distinguish different recharge operations of a same vehicle at a same station. Another option is to develop models based on arc duplication, where each pair of customers is linked by several arcs, each one corresponding to a path visiting only recharge stations; this generates a multi-graph whose size must be taken under control by suitable dominance tests. The two alternatives have been extensively compared by Koyuncu and Yavuz [18]. Notably, the approach investigated in this paper needs neither node duplication nor arc duplication. In principle, a tailored node duplication operation could be needed for some specific station if all branching techniques described in the remainder fail to produce an integer solution, but this never occurred in our computational tests.

3.1 Feasible Paths

We define a *path* to be a sequence of customers visited by the same vehicle between two recharge stations (including the depot). For the sake of clarity, it is worth remarking the difference with respect to path-based models (e.g., Andelmin and Bartolini [1]) where a path is defined as a sequence of recharge stations between two customers.

An "empty path" is a path that directly connects two recharge stations without visiting any customer in between. We indicate with Λ the set of all feasible paths and with $\Lambda_{[u,v]}$ the set of all feasible paths between vertices $u \in \mathcal{R}$ and $v \in \mathcal{R}$; the two endpoints can coincide, because it is allowed for a vehicle to leave a station, to visit some customers and to go back to the same station. We indicate by τ_u the minimum travel time between the depot and vertex u. Assuming P to be an arbitrary subset of vertices, notation $\mathcal{E}_P$ indicates the subset of edges in $\mathcal{E}$ with both endpoints in P. Notation Δ_i indicates the subset of edges in $\mathcal{E}$ with an endpoint in vertex i. Binary variables y_i^l take value 1 if and only if customer $i \in \mathcal{N}$ is visited along path $l \in \Lambda_{[u,v]}$; they are decision variables in the pricing subproblem and fixed coefficients for each column in the master problem. Binary variables z_a^l are edge variables for each edge $a \in \mathcal{E}$ and each path $l \in \Lambda_{[u,v]}$. With this notation, we can now state the formal definition of the set of paths for each pair of stations $[u, v]$.

$$\Lambda_{[u,v]} = \{(y^l, z^l) : \sum_{a\in\Delta_i} z_a^l = 2y_i^l \ \forall i \in \mathcal{N} \tag{1}$$

$$\sum_{a\in\Delta_u} z_a^l = \sum_{a\in\Delta_v} z_a^l = 1 \text{ if } u \neq v \tag{2}$$

$$\sum_{a\in\Delta_u} z_a^l = \sum_{a\in\Delta_v} z_a^l = 2 \text{ if } u = v \tag{3}$$

$$\sum_{a\in\Delta_j} z_a^l = 0 \ \forall j \in \mathcal{R}\backslash\{u, v\} \tag{4}$$

$$\sum_{a \in \mathcal{E}_P} z_a^l \leq |P| - 1 \forall S \subseteq \mathcal{N} \cup \mathcal{R} \backslash \{u, v\},\ P \neq \emptyset \tag{5}$$

$$\sum_{i \in \mathcal{N}} q_i y_i^l \leq Q \tag{6}$$

$$\sum_{a \in \mathcal{E}} t_a z_a^l + \sum_{i \in \mathcal{N}} s_i y_i^l + s_u + s_v + \tau_u + \tau_v \leq T \tag{7}$$

$$\sum_{a \in \mathcal{E}} e_a z_a^l \leq B \tag{8}$$

$$y_i^l \in \{0, 1\} \forall i \in \mathcal{N}$$

$$z_a^l \in \{0, 1, 2\} \forall a \in \mathcal{E}\}. \tag{9}$$

Constraints (1) are degree constraints; constraints () state that u and v must be the endpoint of one selected arc if they are different; constraints (3) state that a self-loop must be incident to a station twice; constraints (4) forbid visits to stations different from u and v; constraints (5) are subtour elimination constraints; constraints (6), (7) and (8) impose limits on the consumption of capacity, time and energy, respectively, on each path independently. According to constraints (9), z_a^l are allowed to take value 2, to include those paths with the same endpoint station and including only one customer.

For each path, that is for each column in the master problem, we also need some additional information. Coefficients q^l indicate the amount of demand served along path l:

$$q^l = \sum_{i \in \mathcal{N}} q_i y_i^l. \tag{10}$$

Coefficients t^l indicate the traveling and service time spent along path l:

$$t^l = \sum_{a \in \mathcal{E}} t_a z_a^l + \sum_{i \in \mathcal{N}} s_i y_i^l + \frac{1}{2}(s_u + s_v). \tag{11}$$

That is, in t^l we account for traveling time and customer service time, and half of the service time in the endpoint stations, as the remaining half is accounted in the adjacent paths while linking them into full routes. Coefficients e^l indicate the energy consumption along path l:

$$e^l = \sum_{a \in \mathcal{E}} e_a z_a^l. \tag{12}$$

Finally c^l indicates the fixed cost for recharge for each path $l \in \Lambda_{[u,v]}$:

$$c^l = f. \tag{13}$$

We associate a binary variable θ_{lk} with each feasible path $l \in \Lambda = \bigcup_{[u,v]} \Lambda_{[u,v]}$ and each vehicle $k \in \mathcal{K}$: θ_{lk} takes value 1 if and only if path l is selected to be part of the solution and is assigned to vehicle k. It is necessary to have as many copies of the path variables as the number of different vehicles in order not to allow capacity, time and energy to be traded between vehicles. Unfortunately this introduces symmetry, i.e., dual degeneracy, in the master problem

3.2 The Master Problem

In our notation w_j^l indicates how many times vertex $j \in \mathcal{R}$ is an endpoint of path $l \in \Lambda$. Extending the notation above, for an arbitrary $S \subseteq \mathcal{R}$, we indicate as Λ_S the set of paths with both endpoints in subset S (that is, $\Lambda_{\mathcal{R}} = \Lambda$).

Suffixes e and ne stand for "empty" and "non-empty," respectively: so, Λ^e (see constraints (27)) is the set of empty paths, while Λ_S^{ne} (see constraints (26)) is the set of non-empty paths with both endpoints in S. Finally, $\mathcal{L}_S$ indicates the set of paths with one endpoint in subset S.

Each continuous non-negative variable δ_{jhk} indicates the amount of energy recharged by vehicle $k \in \mathcal{K}$ at station $j \in \mathcal{R}$ with technology $h \in \mathcal{H}_j$. Each discrete variable ω_{jk} indicates how many times vehicle $k \in \mathcal{K}$ visits station $j \in \mathcal{R}$.

The master problem reads as follows.

$$\text{minimize} \sum_{l \in \Lambda} \sum_{k \in \mathcal{K}} c^l \theta_{lk} + \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{R}} \sum_{h \in \mathcal{H}_j} \gamma_h \delta_{jhk} \tag{14}$$

$$\text{s.t.} \sum_{l \in \Lambda} w_j^l \theta_{lk} = 2\omega_{jk} \ \forall j \in \mathcal{R}, \forall k \in \mathcal{K} \tag{15}$$

$$\omega_{0k} \leq 1 \ \forall k \in \mathcal{K} \tag{16}$$

$$\sum_{l \in \mathcal{L}_S} \theta_{lk} \geq 2 \sum_{l' \in \P_S^{ne}} y_i^{l'} \theta_{l'k} \ \forall S \subseteq \mathcal{R} \setminus \{0\}, \ \forall i \in \mathcal{N}, \ \forall k \in \mathcal{K} \tag{17}$$

$$\sum_{k \in \mathcal{K}} \sum_{l \in \Lambda} y_i^l \theta_{lk} \geq 1 \ \forall i \in \mathcal{N} \tag{18}$$

$$\sum_{l \in \Lambda} q^l \theta_{lk} \leq Q \ \forall k \in \mathcal{K} \tag{19}$$

$$\sum_{l \in \Lambda} t^l \theta_{lk} + \sum_{j \in \mathcal{R} \setminus \{0\}} \sum_{h \in \mathcal{H}_j} \frac{\delta_{jhk}}{\rho_h} \leq T \ \forall k \in \mathcal{K} \tag{20}$$

$$\sum_{l \in \mathcal{L}_S \cup \P_S} e^l \theta_{lk} - \sum_{j \in S} \sum_{h \in \mathcal{H}_j} \delta_{jhk} \leq \frac{1}{2} B \sum_{l \in \mathcal{L}_S} \theta_{lk} \ \forall S \subseteq \mathcal{R}, \ \forall k \in \mathcal{K} \tag{21}$$

$$\sum_{j \in S} \sum_{h \in \mathcal{H}_j} \delta_{jhk} - \sum_{l \in \P_S} e^l \theta_{lk} \leq \frac{1}{2} B \sum_{l \in \mathcal{L}_S} \theta_{lk} \ \forall S \subseteq \mathcal{R}, \ \forall k \in \mathcal{K} \tag{22}$$

$$\omega_{0k} \leq \omega_{0\ k-1} \ \forall k \in \mathcal{K} \tag{23}$$

$$\sum_{j \in \mathcal{R}} \omega_{jk} \leq \sum_{j \in \mathcal{R}} \omega_{j\ k-1} \ \forall k \in \mathcal{K} \tag{24}$$

$$\delta_{jhk} \geq 0 \ \forall j \in \mathcal{R}, \forall h \in \mathcal{H}_j, \forall k \in \mathcal{K} \tag{25}$$

$$\theta_{lk} \in \{0, 1\} \ \forall l \in \Lambda^{ne}, \ \forall k \in \mathcal{K} \tag{26}$$

$$\theta_{lk} \text{ integer } \forall l \in \Lambda^{e}, \ \forall k \in \mathcal{K} \tag{27}$$

$$\omega_{jk} \text{ integer } \forall j \in \mathcal{R}, \forall k \in \mathcal{K}. \tag{28}$$

Degree constraints (15) impose that each station vertex has even degree, i.e. that the edge variables θ define an Euler graph. Constraints (16) state that each vehicle must visit the depot once. Constraints (17) are sub-tour elimination constraints, as illustrated in Fig. 1. Including coefficients y_i^l allows us to guarantee the sum on the right-hand side to always be binary, and therefore to have only one constraint aggregating all the elements of $\P_S^{ne}$. Their number is therefore not exponential in the

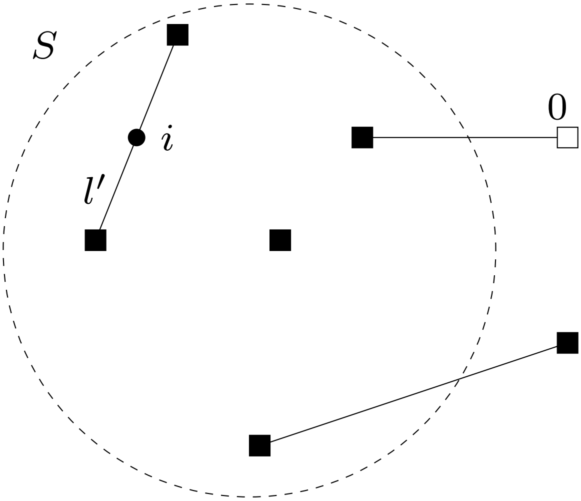

Fig. 1 Structure of subtour elimination constraints. Given any subset S of recharge stations, not including the depot 0, if a solution contains a non-empty path l' with both endpoints in S ($l' \in \P_S^{ne}$), then it must also contain at least two paths connecting stations in S with stations not in S (paths in $\mathcal{L}_S$)

number of customers, but only in the number of stations and technologies: they can be generated either when needed or since the beginning, depending on the size of the instance. Constraints (18) are covering constraints, stating that each customer must be visited. Constraints (19) and (20) are resource constraints on capacity and time.

Constraints (21) and (22) impose lower and upper bounds to the amounts of energy recharged in each subset of stations. Intuitively, constraints (21) have the following meaning. Every time a vehicle visits a subset $S \in \mathcal{R}$ it is allowed to enter it with full battery and leave with empty battery; hence, the difference between the amount of energy consumed to travel to, within, and from subset S (first term of the left hand side) and the total amount of energy recharged within S (second term on the left hand side) is upper bounded by the battery capacity B multiplied by the number of times the vehicle enter and leaves S. Symmetrically, constraints (22) impose an upper bound on the difference between the total recharge and the total consumption, because of the limited battery capacity. A formal proof of the validity of constraints (21) and (22) is given in [5].

Inequalities (23) and (24) are symmetry breaking constraints. Finally constraints (25) are non-negativity conditions on variables δ, constraints (26) and (27) are integrality restrictions on variables θ and constraints (28) are integrality restrictions on variables ω.

Remark. The formulation given above allows for convex combinations of recharges with different technologies in a same station and aggregates the overall amount of energy recharged by the same vehicle in the same station using the same technology over all its visits. The rationale for accepting this relaxation when solving the master problem is that a convex combination of two technologies can belong to an optimal recharge policy along a given route only when time constraints are binding and the value of the maximum route duration falls in a range which is smaller than $|B(\rho' - \rho'')|$, where ρ' and ρ'' indicate the recharge speed of the two technologies; this is unlikely to happen in randomly generated instances. The constraint that forbids convex combinations of recharges is enforced incrementally through a suitable branching technique, fully detailed in Section 4.

3.2.1 Reduced Costs

From the constraints of the master problem we can obtain the expression of the reduced costs $\overline{c}_{lk}$ for each path $l \in \Lambda_{uv}$ connecting stations $u \in \mathcal{R}$ and $v \in \mathcal{R}$ and for each vehicle $k \in \mathcal{K}$. We indicate the dual variables with the symbol $\beta^{(n)}$, where n is the index of the corresponding constraint set in the master problem. We assume that all inequality constraints in the master problem have been written in $\geq$ form, so that their corresponding dual variables are non-negative. The reduced cost of each column corresponding to path $l \in \Lambda_{[u,v]}$ and vehicle $k \in \mathcal{K}$ is given by the following expression:

$$\begin{aligned}\bar{c}_{lk} = c^l - \sum_{j\in\mathcal{R}} w_j^l \beta_{jk}^{(15)} - \sum_{S\subseteq\mathcal{R}\setminus\{0\}:l\in\mathcal{L}_S} \sum_{l'\in\P_S^{ne}} \sum_{i\in\mathcal{N}} \beta_{Sik}^{(17)} + 2 \sum_{S\subseteq\mathcal{R}\setminus\{0\}:l\in\P_S^{ne}} \sum_{i\in\mathcal{N}} y_i^l \beta_{Sik}^{(17)} \\ - \sum_{i\in\mathcal{N}} y_i^l \beta_i^{(18)} + q^l \beta_k^{(19)} + t^l \beta_k^{(20)} \\ - \frac{1}{2} B \sum_{S:l\in\mathcal{L}_S} \beta_{Sk}^{(21)} + e^l \sum_{S:l\in\mathcal{L}_S\cup\P_S} \beta_{Sk}^{(21)} - \frac{1}{2} B \sum_{S:l\in\mathcal{L}_S} \beta_{Sk}^{(22)} - e^l \sum_{S:l\in\P_S} \beta_{Sk}^{(22)}.\end{aligned}$$

We now replace q^l, t^l, e^l and c^l with their definitions (10)-(13) and then we group all terms that only depend on u, v and k, and not on y and z variables, in a unique term

$$\begin{aligned}\sigma_{uvk} =& f - \sum_{j\in\mathcal{R}} w_j^l \beta_{jk}^{(15)} - \sum_{S\subseteq\mathcal{R}\setminus\{0\}:l\in\mathcal{L}_S} \sum_{l'\in\P_S^{ne}} \sum_{i\in\mathcal{N}} \beta_{Sik}^{(17)} \\ &+ \frac{1}{2}(s_u + s_v)\beta_k^{(20)} - \frac{1}{2} B \sum_{S:l\in\mathcal{L}_S} (\beta_k^{(21)} + \beta_k^{(22)})\end{aligned}$$

which is a constant for each given pair $[u, v]$ and each vehicle k. Now the objective function of the pricing problem for each $l \in \Lambda_{[u,v]}$ and each vehicle $k \in \mathcal{K}$ can be restated as follows.

$$\begin{aligned}\bar{c}_{lk} = \sigma_{uvk} + 2 \sum_{S\subseteq\mathcal{R}\setminus\{0\}:l\in\P_S^{ne}} \sum_{i\in\mathcal{N}} y_i^l \beta_{Sik}^{(17)} - \sum_{i\in\mathcal{N}} y_i^l \beta_i^{(18)} + \sum_{i\in\mathcal{N}} q_i y_i^l \beta_k^{(19)} \\ + \sum_{a\in\mathcal{E}} t_a z_a^l \beta_k^{(20)} + \sum_{i\in\mathcal{N}} s_i y_i^l \beta_k^{(20)} + \sum_{S:l\in\mathcal{L}_S} \sum_{a\in\mathcal{E}} e_a z_a^l \beta_{Sk}^{(21)} + \sum_{S:l\in\P_S} \sum_{a\in\mathcal{E}} e_a z_a^l (\beta_{Sk}^{(21)} - \beta_{Sk}^{(22)})\end{aligned}$$

The objective function can be rewritten, grouping three different terms: fixed costs, vertex costs and edge costs.

$$\begin{aligned}\bar{c}_{lk} = \sigma_{uvk} + \sum_{i\in\mathcal{N}} \left(2 \sum_{S\subseteq\mathcal{R}\setminus\{0\}:l\in\P_S^{ne}} \beta_{Sik}^{(17)} - \beta_i^{(18)} + q_i \beta_k^{(19)} + s_i \beta_k^{(20)} \right) y_i^l \\ + \sum_{a\in\mathcal{E}} (t_a \beta_k^{(20)} + e_a(\sum_{S:l\in\mathcal{L}_S\cup\P_S} \beta_{Sk}^{(21)} - \sum_{S:l\in\P_S} \beta_{Sk}^{(22)}) z_a^l.\end{aligned}$$

Now we define:

$$c_{iuvk}^{vertex} = 2 \sum_{S\subseteq\mathcal{R}\setminus\{0\}:u\in S\wedge v\in S} \beta_{Sik}^{(17)} - \beta_i^{(18)} + q_i \beta_k^{(19)} + s_i \beta_k^{(20)}$$

$$c_{auvk}^{edge} = t_a \beta_k^{(20)} + e_a(\sum_{S:u\in S\vee v\in S} \beta_{Sk}^{(21)} - \sum_{S:u\in S\wedge v\in S} \beta_{Sk}^{(22)})$$

and we obtain, for each $l \in \Lambda_{[u,v]}$ and each vehicle $k \in \mathcal{K}$

$$\bar{c}_{lk} = \sigma_{uvk} + \sum_{i\in\mathcal{N}} c_{iuvk}^{vertex} y_i^l + \sum_{a\in\mathcal{E}} c_{auvk}^{edge} z_a^l. \tag{29}$$

3.3 The Pricing Sub-Problem

From the definition of path (1) - (8) and owing to the algebraic manipulations described in the previous subsection, the pricing sub-problem is formulated as follows.

$$\text{minimize } \bar{c}_{lk} = \sigma_{uvk} + \sum_{i \in \mathcal{N}} c_{ik}^{vertex} y_i + \sum_{a \in \mathcal{E}} c_{ak}^{edge} z_a \tag{30}$$

$$\text{s.t. } \sum_{a \in \Delta_i} z_a = 2y_i \ \forall i \in \mathcal{N} \tag{31}$$

$$\sum_{a \in \Delta_u} z_a = \sum_{a \in \Delta_v} z_a = 1 \text{ if } u \neq v \tag{32}$$

$$\sum_{a \in \Delta_u} z_a = \sum_{a \in \Delta_v} z_a = 2 \text{ if } u = v \tag{33}$$

$$\sum_{a \in \Delta_j} z_a = 0 \ \forall j \in \mathcal{R} \backslash \{u, v\} \tag{34}$$

$$\sum_{a \in \mathcal{E}_S} z_a \leq |S| - 1 \ \forall S \subseteq \mathcal{N} \cup \mathcal{R} \backslash \{u, v\}, \ S \neq \emptyset \tag{35}$$

$$\sum_{a \in \mathcal{E}} e_a z_a^l \leq B \tag{36}$$

$$\sum_{i \in \mathcal{N}} q_i y_i \leq Q \tag{37}$$

$$\sum_{a \in \mathcal{E}} t_a z_a + \sum_{i \in \mathcal{N}} s_i y_i + s_u + s_v + \tau_u + \tau_v \leq T \tag{38}$$

$$y_i \in \{0, 1\} \ \forall i \in \mathcal{N} \tag{39}$$

$$z_a \in \{0, 1\} \ \forall a \in \mathcal{E}. \tag{40}$$

Pricing variables y_i (resp. z_a) are encoded as master coefficients y_i^l (resp. z_a^l) once an optimal path is found.

3.3.1 An Exact Pricing Algorithm

The pricing sub-problem is solved to optimality for each $u, v \in \mathcal{R}$ and each $k \in \mathcal{K}$ by a bi-directional dynamic programming algorithm, where states correspond to partial paths. One main advantage of the path-based formulation is that recharge stations are not visited along the paths and hence we do not have to take into account the possible outcomes of recharge operations. Therefore the pricing sub-problem is merely combinatorial.

State. The state contains the following pieces of information:

- the last customer vertex $i \in \mathcal{N}$ that has been reached by the partial path;
- the subset $S \subseteq \mathcal{N}$ of customer vertices that have already been visited along the partial path;
- the maximum residual capacity η that can be available after the operations at vertex i;
- the maximum residual time t that can be available after the operations at vertex i;
- the maximum residual amount of energy e that can be available after the operations at vertex i;
- the reduced cost $\overline{c}$ of the partial path.

Initialization and termination. The initial empty partial path is represented by the following states: $(u, \emptyset, Q, T - s_u, B, \sigma_{uvk})$ and $(v, \emptyset, Q, T - s_v, B, \sigma_{uvk})$. Bi-directional extension of labels terminates when half of a critical resource has been used. In the case of paths, it is reasonable to assume that energy is the binding resource; hence extensions are stopped when the value of e for the resulting labels would be smaller than $B/2$.

Extension. Consider a state $(i, S', \eta', t', e', \overline{c}')$ associated with vertex i; when it is extended by appending an additional edge $[i, j]$ to the partial path, it produces one or more states of the form $(j, S'', \eta'', t'', e'', \overline{c}'')$. Extensions to recharge stations are not allowed; since node $j \in \mathcal{N}$ is a customer vertex, the resource extension function is as follows:

$$\begin{aligned}
&S'' = S' \cup \{j\}\\
&\eta'' = \eta' - q_j\\
&t'' = t' - t_{ij} - s_j\\
&e'' = e' - e_{ij}\\
&\overline{c}'' = \overline{c}' + c_{jk}^{vertex} + c_{ijk}^{edge}
\end{aligned}$$

Feasibility constraints. Not all states generated in this way are feasible. In particular they must comply with the following constraints:

$$\begin{aligned}
&j \notin S'\\
&\eta'' \geq 0\\
&(e'' \geq 0)\\
&t'' \geq 0
\end{aligned}$$

Feasibility condition $e'' \geq 0$ is not checked because it is dominated by the termination condition $e'' \geq B/2$ which stops the bi-directional extension. Arcs (i,j) having $e_{ij} > B/2$ are still considered during join.

Bounding. For each state generated by the dynamic programming algorithm we also compute a bound to the reduced cost that can be achieved, in order to early detect states that cannot lead to negative reduced cost solutions and to discard them from further consideration. For this purpose a set of q-routes is precomputed for each station. In our case, a q-route is a minimum reduced cost path from any vertex i to the destination station requiring the use of at most q energy units. Such a path is not required to satisfy either time constraints or elementarity conditions; therefore, the set of q-routes for all integer values of q between 0 and B and for each vertex i can be computed in pseudo-polynomial time [8] once for each column generation iteration, before starting the pricing algorithm. Let $\psi(i,q)$ be the costs of q-routes in vertex i and let $(i,S,\eta,t,e,\bar{c})$ be a dynamic programming state generated in vertex i. If $\bar{c} + \psi(i,e) \geq 0$, then the state can be discarded as no feasible extension can yield a negative reduced cost path. States are then evaluated (and possibly discarded) according to the best reduced cost that can be achieved with the corresponding residual amount of energy.

Dominance. Given two states $(i,S',\eta',t',e',\overline{c}')$ and $(i,S'',\eta'',t'',e'',\overline{c}'')$ associated with the same vertex i, the former dominates the latter only if all these conditions hold and at least one of the inequalities is strict.

$$\begin{aligned}
&S' \cup U' \subseteq S'' \cup U'' \\
&\eta' \geq \eta'' \\
&t' \geq t'' \\
&e' \geq e'' \\
&\overline{c}' \leq \overline{c}''.
\end{aligned}$$

Sets U' and U'' represent the sets of unreachable vertices, following the idea described in [12].

Join. When two partial paths, originating from $u \in \mathcal{R}$ and $v \in \mathcal{R}$ (where u and v may well be the same recharge station), are joined together to produce a complete path, the following feasibility tests are done on the two corresponding labels $(i,S',\eta',t',e',\overline{c}')$ and $(j,S'',\eta'',t'',e'',\overline{c}'')$:

$$\begin{aligned}
&S' \cap S'' = \emptyset \\
&(Q-\eta') + (Q-\eta'') \leq Q \\
&(T-t') + (T-t'') + t_{ij} \leq T \\
&(B-e') + (B-e'') + e_{ij} \leq B
\end{aligned}$$

The reduced cost of the resulting path is

$$\overline{c} = \overline{c}' + \overline{c}'' + c^{edge}_{ijk}.$$

The Join phase goes on until a feasible (acyclic) path is found with negative reduced cost. As soon as such a path is found, a corresponding column is inserted into the master problem and the Join step ends. At the contrary, if no path is found with negative reduced cost during Join, pricing stops.

To speed up the pricing algorithm we proceed as follows. First, the set of partial paths originating from each station is computed independently. Then all pairs of stations are considered, and the corresponding partial paths are tentatively joined. We remark that, due to the contribution of the subtour elimination constraints dual variables, the same path can have different reduced cost for different pairs of stations. Therefore, we delay the reduced cost computation of each path at this stage.

Speedup techniques. It is well-known that the main source of complexity when routes or paths are priced out is the combinatorial explosion due to the subset S in the state. To cope with this problem, several techniques have been proposed. We use the so-called *ng*-routes [3], i.e. we replace the test $S' \cup U' \subseteq S'' \cup U''$ on the set of visited vertices with a similar test on a smaller subset of vertices: we indicate with NG_i for each vertex i the vertex subset including i and its k nearest neighbors, according to the reduced edge costs (after some tests we set $k = \min\{\max\{N/10, 5\}, N\}$). The dominance test is then $(S' \cup U') \cap NG_i \subseteq (S'' \cup U'') \cap NG_i$. Extension operations are modified as well, setting $S = S \cap NG_i$ whenever a new label is created at node i.

We also employ a decremental state-space relaxation technique: due to the NG-route relaxation, the minimum reduced cost path obtained during join may contain cycles. If it is the case, and if the reduced cost of such a path is negative, then the NG subsets of the vertices in the cycles are enlarged accordingly and the pricing algorithm is restarted.

Capacity constraint relaxation. In our final implementation we decided to disregard the capacity constraint in the pricing problem, because it is unlikely to be binding in a single path. This implies weakening the relaxation of the master problem, but it reduces the computation time significantly.

3.3.2 Heuristic Pricing

Before running the exact pricing algorithm, columns with negative reduced cost are searched by two heuristic pricing algorithms. The first one is a nearest neighbor greedy algorithm. We run it for each vehicle and each pair of starting and ending stations. Beginning with the starting station, the vertex not belonging to the path, which can still be visited without violating resource limits, and leading to the largest reduction in the reduced cost, is selected. If no such a vertex can be found, the path is closed by visiting the ending station. Otherwise the selected vertex is added to the path and the process is iterated from that vertex. The second one is a heuristic version of the exact pricing algorithm, with two main modifications: (i) from each vertex the extension is done only to the three closest vertices; (ii) although visiting an already visited vertex is forbidden, the dominance test does not check the set of visited nodes. Therefore the algorithm is faster but it may happen that the optimal solution be missed or be dominated by a sub-optimal one.

4 Branch-and-Cut-and-Price

Owing to the characteristics of the path-based formulation, it is quite important to devise an effective implicit enumeration scheme, embedding suitable disaggregation schemes, branching rules and cutting planes. In fact, we need to cope with optimal solutions of the master problem potentially made by many paths combined in a fractional way, i.e. very far from complying with the integrality requirements.

In the remainder we describe the techniques to enforce integrality and the cutting planes strategies we have used in the final version of our algorithm. They were designed by exploiting the combinatorial structure of the problem, but finally chosen after extensive preliminary computational tests and analysis.

4.1 Incremental Variable Disaggregation

As discussed above, constraints on the mutually exclusive choice of the recharge technology are initially relaxed by aggregation, thus allowing for linear combinations of them. To guarantee feasibility, we designed an ad-hoc progressive strengthening technique, that we name *incremental disaggegation.* A set of integer variables ω_{hjk} is introduced, representing the number of times vehicle k visits station j using technology h; these new variables are linked by the following set of constraints:

$$\delta_{jhk} \leq B \cdot \omega_{jhk} \;\; \forall j \in \mathcal{R}, h \in \mathcal{H}_j, k \in \mathcal{K}.$$

Variables x_{jhk} are then linked to variables ω_{jk} as follows:

$$\sum_{h \in \mathcal{H}_j} \omega_{jhk} = \omega_{jk} \;\; \forall j \in \mathcal{R}, k \in \mathcal{K}.$$

These constraints have no effect when integrality conditions are dropped, but they impose tight restrictions when ω_{jk} and ω_{jhk} variables are fixed by branching decisions (see branching rules 2 and 7 below).

Fixing these variables to integer values is however not enough to fully forbid recharge operations made by the same vehicle which are inconsistent with its final route. In particular, two *inconsistency conditions* might still arise: recharges with different technologies are mixed during a single visit to a station, and recharges with the same technology are mixed during multiple visits to the same station.

Therefore, we additionally design an *on-demand* node duplication rule, which is triggered only when inconsistencies are detected in a fractional solution due to these specific aggregation cases.

In detail, when we detect that a vehicle visiting a station more than once does mix recharges with different technologies during the same stop, we replicate the station node into as many copies as the number of technologies that are available at the station, making a single technology available in each replica, and we resume the optimization process. A similar technique is applied when we detect that a vehicle visits a station more than once using the same technology in different stops. In this case the station node is replicated, and a single visit is allowed to each replica.

4.2 Branching Rules

We employ seven branching rules, all originating binary branches.

Branching rule 1: Number of paths. We branch by imposing that the overall number of paths in the solution is upper bounded by m in one branch and lower bounded by $m + 1$ in the other for a suitably chosen integer value m.

Branching rule 2: Number of visits. We branch by imposing that a certain vehicle visits a certain station at most m times in one branch and at least $m + 1$ times in the other.

Branching rule 3: Empty paths. We branch by imposing that a certain empty path is used at most m times in one branch and at least $m + 1$ times in the other.

Branching rule 4: Vertex-path assignment. We branch by imposing that a certain customer vertex is visited or not along a path connecting two suitably chosen stations.

Branching rule 5: Edges. We branch on the use of a certain edge of the graph.

Branching rule 6: Customer-vehicle assignment. We branch by imposing that a customer vertex is served by a vehicle in a certain vehicle subset or in its complement.

Branching rule 7: Choice of recharge technology during single visits.

We branch by imposing that a certain vehicle that visits a certain station only once uses a specific technology or not.

We experimented on many branching rule selection policies. We found the following one to work best:

- if the overall number of paths in the solution is fractional, apply branching rule 1;
- else if a vehicle visits a station a fractional number of times $\bar{m}$ with $\min\{\bar{m} - \lfloor \bar{m} \rfloor, \lceil \bar{m} \rceil - \bar{m}\} > 10^{-3}$, apply branching rule 2;
- else if an empty path is used a fractional number of times $\bar{m}$ with $\min\{\bar{m} - \lfloor \bar{m} \rfloor, \lceil \bar{m} \rceil - \bar{m}\} > 10^{-3}$, apply branching rule 3;
- else if a customer vertex is visited a fractional number $\bar{m}$ of times along a path connecting two stations with $\min\{\bar{m}, 1 - \bar{m}\} > 10^{-1}$, apply branching rule 4;
- else if an edge of the graph is fractionally used $\bar{m}$ times with $\min\{\bar{m}, 1 - \bar{m}\} > 10^{-1}$, apply branching rule 5;
- else if a vehicle visits a station a fractional number of times $\bar{m}$ (regardless of the value of $\bar{m}$), apply branching rule 2;
- else if an empty path is used a fractional number of times $\bar{m}$ (regardless of the value of $\bar{m}$), apply branching rule 3;
- else if a customer vertex is visited a fractional number $\bar{m}$ of times along a path (regardless of the value of $\bar{m}$), apply branching rule 4;
- else if an edge of the graph is used a fractional number $\bar{m}$ of times (regardless of the value of $\bar{m}$), apply branching rule 5;

- else if a customer vertex is fractionally served by vehicles in a vehicle subset, apply branching rule 6;
- else if a vehicle visits a station once, fractionally using different recharge technologies, apply branching rule 7;
- otherwise, trigger the incremental variable disaggregation procedure.

Hence, the branching rules are applied in cascade, but rules 2, 3, 4 and 5 are initially skipped if the corresponding branching variables are not "fractional enough"; they are however given a second chance for branching before considering rules 6 and 7. Such a choice helps in avoiding poor bounds improvement after branching on almost integral variables. Indeed it matches a common intuition in combinatorial optimization algorithms: to make coarse decisions earlier (like the overall number of paths in a solution, or the set of stations visited by each vehicle), and fine-grained decisions later (like the use of single edges).

The incremental variable disaggregation procedure is integrated as follows. First, we check if any vehicle visits a station twice or more, and one of the inconsistency conditions of Section 4.1 occurs. If it is the case, incremental variable disaggregation is performed, and the optimization process is resumed after replacing the branching sub-problem with its disaggregated version. Otherwise, the solution is integer and feasible, and the sub-problem is fathomed, possibly updating the primal bound.

Only in two cases (namely instances B-C4-N030 and C-24-N10), we observed that solutions produced during the branch-and-cut-and-price tree after the application of branching rules 1-6 included a recharge operation that was the convex combination of two recharges with different technologies at the same station. However, branching rule 7, which is specific for the EVRP with multiple technologies, was enough to eliminate the infeasiblity.

Incremental disaggregation is kept as a last chance, since it implies enlarging the graph, adding substantial burden in terms of both master problem size and number of pricing problems to solve. In principle it may be necessary to achieve integrality; however, in our implementation, when the optimal solution of the master problem in the current branch-and-bound sub-problem would require node duplication according to incremental variable disaggregation, we assign the sub-problem a very low priority in the list of open sub-problems in the branch-and-bound tree, in order to delay the time-consuming analysis of its enlarged version as much as possible. In our computational tests, this allowed to never actually explore such branch-and-bound sub-problems, since subsequent improvements in the bounds allowed to discard them before any node duplication.

4.3 Cutting Planes

We made computational tests with several different combinations of dynamic separation strategies. The best performing combination was eventually the following.

Sub-tour elimination constraints

$$\sum_{l \in \mathcal{L}_S} \theta_{lk} \geq 2 \sum_{l' \in \P_S^{ne}} y_i^{l'} \theta_{l'k} \quad \forall S \subseteq \mathcal{R} \backslash \{0\},\ \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}$$

are exponential in the number of stations, but, since their number is not so large, we could include all these cuts in the master problem since the beginning.

Energy consumption constraints

$$\sum_{l \in \mathcal{L}_S \cup \P_S} e^l \theta_{lk} - \sum_{j \in S} \sum_{h \in \mathcal{H}_j} \delta_{jhk} \leq \frac{1}{2} B \sum_{l \in \mathcal{L}_S} \theta_{lk} \quad \forall S \subseteq \mathcal{R},\ \forall k \in \mathcal{K}$$

$$\sum_{j \in S} \sum_{h \in \mathcal{H}_j} \delta_{jhk} - \sum_{l \in \P_S} e^l \theta_{lk} \leq \frac{1}{2} B \sum_{l \in \mathcal{L}_S} \theta_{lk} \quad \forall S \subseteq \mathcal{R},\ \forall k \in \mathcal{K}$$

are exponential in the number of stations and they are not included in the initial restricted master problem; instead, they are dynamically separated by the branch-and-cut-and-price framework that we employed [26].

The formulation of the master problem was also strengthened by capacity cuts. Given a vertex subset S such that at least m vehicles are required to satisfy its overall demand, we impose

$$\sum_{l \in \mathcal{L}_S} \theta_l \geq 2m$$

that is, at least $2m$ paths must connect the subset to the rest of the graph in any feasible solution.

These constraints are treated in their weak form by relaxing them as master constraints, in order not to modify the pricing sub-problem. These capacity constraints are exponential in the number of vertices and they are not included in the initial restricted master problem. To separate these constraints we use heuristics included in Lysgaard's library [20]. We consider an auxiliary graph, in which a flow is associated with each edge, representing how much the edge is used in the current fractional solution. Lysgaard's heuristics [20] assume to work on a graph with a total flow equal to 2 on the edges incident to each vertex; this is not guaranteed in our model. Hence we define an auxiliary graph as follows. First, we compute the total flow $\tilde{f}_j$ on the edges incident to each vertex $j \in \mathcal{R}$, and we replace vertex j with $\lceil \tilde{f}_j/2 \rceil$ copies of it; accordingly, we replace each edge incident to j with $\lceil \tilde{f}_j/2 \rceil$ copies of it and we uniformly split the corresponding flow among those edges. We remark that, as a result of this splitting, also some vertex $i \in \mathcal{N}$ might have less than 2 units of total flow on incident edges. Then we add self-loops to each vertex $i \in \mathcal{N} \cup \mathcal{R}$ and we assign the self-loops the amount of flow needed to reach a total of 2 for each vertex. Finally, we run Lysgaard's heuristics on the resulting graph. Upon completion, we insert into the restricted master problem each cut that was found in this way and whose violation is at least 10^{-4}. For computational reasons, when a vertex is found to have $\lceil \tilde{f}_j/2 \rceil > |\mathcal{N}|/2$, the separation of capacity cuts is not performed.

The separation of new cuts is done only when column generation is over. The whole pricing and cutting loop is repeated until neither violated cuts nor improving columns are found.

4.4 Primal Heuristics

Feasible solutions are computed through the column generation heuristics implemented in SCIP 6.0. The primal values obtained in this way are reported in columns CGH in the tables. Furthermore, once the master problem has been optimized at the root node, we also compute a feasible solution by running CPLEX on the master problem with integrality constraints, including all the columns previously computed. The results obtained in this way are reported in the columns MIPH in the tables.

4.5 Parallelization

As reported above, an appealing feature of path-based formulations is the possibility of performing multiple pricing in parallel. In our case a set of $K \cdot R \cdot (R-1)/2$ pricing sub-problems, one for each pair of stations and each vehicle, can be solved in parallel. From an algorithmic point of view, however, such a massive parallelism can be exploited only if (a) enough physical computing resources are available and (b) a suitable set of threads can be created, along with proper data structures making them disjoint or allowing them to be synchronized with very limited waiting times. Indeed, finding good parallelization schemes in column generation is not trivial [4], as bottleneck effects can often be observed.

We made experiments with different synchronization techniques, finding the following one to produce the best results.

First, we run for each triple (starting station, ending station, vehicle) fast heuristic pricing algorithms in parallel; the best solution for each triple is kept as an estimate of the optimal reduced cost in the corresponding pricing problem. Then we sort the set of triples by non-decreasing values of these estimates and we split the sorted list into blocks of H elements each. Finally, we sequentially examine the blocks, running up to H exact pricing algorithms in parallel for each block. Furthermore, extension operations in the dynamic programming algorithm are always run in parallel. If, at the end of a block computation, columns with negative reduced cost are found, these are added to the master problem and the pricing process is stopped; otherwise another block is processed.

In our tests the value of H was set to the number of hardware threads supported by our PC, i.e 32 (but each of them was running up to N threads in parallel during extension).

From the data structures point of view, we found useful to trade memory for computing time, duplicating support data when needed, to reduce thread synchronization needs. However we could not make threads fully disjoint, because the access for writing the inner SCIP data structures requires synchronization.

5 Computational Experiments

Our algorithms were implemented in C++, using SCIP 6.0 [26] as a branch-and-cut-and-price framework. Our version of SCIP embeds CPLEX 12.8 with default settings as an LP solver. We experimented on introducing stabilization techniques and using the barrier algorithm to solve the linear restricted master problem. In both cases we observed a reduction on the number of column generation iterations needed to converge, but the overall performance improvement was negligible. This is because stabilization and the barrier method help to reduce the number of initial and useless column generation iterations, but in our case most of the CPU time was spent during few final column generation iterations, when the exact dynamic programming algorithm is called. The use of our pricing heuristics can be considered itself as a useful tool to overcome stability problems.

The results reported in this section were obtained using a PC equipped with an AMD 1950x 4.0GHz processor and 32 GB of RAM, running Linux Ubuntu 18. The CPU has 16 cores: unless otherwise indicated, computing times are expressed as execution (clock) times. Parallelization was implemented with the multi-threading framework of OPEN-MP.

5.1 Datasets

We tested our algorithms on three datasets.

Dataset A, shown in Table 5 of Appendix, was derived from the Solomon dataset by Schneider, Stenger and Goeke [24], 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. For some instances in this dataset we also modified the number of vehicles with respect to the original value used in [24]. In one case this was done to make the instance feasible, because the original one was not [25]. In some other cases we decreased the number of vehicles to the minimum value for which the instance was known to be feasible [13]. In Tables 5, 6 and 7 R_0 indicates the number of stations (excluding the depot) while T indicates the number of technologies (when it is two or more).

The models of [24] and [13] assume that all vehicles start with fully charged batteries, because it is common to recharge electric vehicles during the night and because current technology already allows to do this with normal power supply means, provided that enough time is available. This is always convenient, since it makes it possible to recharge at the lowest cost. Our methods allow to treat the night recharge as any other recharge operation: to make our experiments consistent with those of [24] and [13], in dataset A we gave the depot a dummy technology (which can be used for the initial recharge) being arbitrary quick and cheap.

Furthermore, to show the need of explicitly handling the presence of multiple technologies, as well as the impact of the free overnight recharge, we have modified a subset of instances from dataset A. The details of this study are reported in subsection 5.2.1.

We also used two more datasets, also considered in [13]. Dataset B, described in Table 6 of Appendix, is adapted from the Solomon dataset: all instances have 30 customers, 7 vehicles, 5 stations and 3 technologies. In this dataset customer locations are clustered.

In Dataset C, described in Table 7 of Appendix, instances have 10 customers, up to 5 vehicles, up to 9 stations and 3 technologies.

5.2 Computational Results

In this subsection we present some computational results obtained with our branch-and-cut-and-price algorithm.

In particular subsubsection 5.2.1 shows how the optimal solution of the EVRP can be significantly different when multiple technologies are taken into account with respect to the version with a single technology.

Subsubsection 5.2.2 aims at evaluating the effect of capacity cuts in strengthening the lower bound.

Subsubsection 5.2.3 shows a comparison between heuristics CGH and MIPH and the local search heuristic 48A from the literature [13].

Subsubsection 5.2.4 presents an evaluation of the speed-up effects obtained through parallelization.

Finally, subsubsection 5.2.5 presents the results of the branch-and-cut-and-price algorithm on the three datasets.

5.2.1 Single vs. Multiple Technologies

Many characteristics of the optimal solutions may change when multiple technologies are introduced. We modified some instances taken from dataset A, to prove that the presence of different technologies, as well as the availability of free overnight recharge, may actually change the structure of optimal solutions.

Table 1 summarizes the comparison between several features of optimal solutions, listed one for each row, in three different cases: ORIG refers to the case where no free charge is available at the starting depot, so that the initial charge is paid as all the others; ST refers to the case with free initial charge at the depot and a single technology at the stations; MT refers to the case with free initial charge and multiple technologies. We remark that our algorithm is able to give optimality guarantees in all these cases by a simple proper encoding of data. The table reports the relative gain of ST over ORIG, MT over ORIG, and MT over ST, as indicated in the leading row. The results refer to five instances: C101-5, C103-5, R104-5, C103-15 and C202-15. They contain either 5 or 15 customers, as indicated in the instance name. The results indicate clearly that these modeling choices have very high impact on several features of the optimal solutions. The effect that can be seen in small instances can be even more remarkable in large ones.

Table 1 Mean absolute percentage difference between solutions using different models

Feature	|ORIG-ST|/ORIG	|ORIG-MT|/ORIG	|ST-MT|/ST
Optimal solution value	56.66%	55.40%	7.94%
Number of paths	10.67%	8.00%	9.00%
Number of empty paths	20.00%	0.00%	20.00%
Customers in the longest path	8.33%	6.67%	10.67%
Number of partial recharges	26.67%	31.67%	21.67%
Number of used stations	5.00%	15.00%	21.67%
Number of used vehicles	40.00%	40.00%	0.00%
Computing time	464.24%	371.17%	15.98%

As an example, the optimal solutions of instance C101-5, for the ORIG, ST and MT cases, are shown in Fig. 2. The instance involves two available vehicles, five customers (vertices labeled 1-5), and three recharge stations, the central one acting also as depot (vertices R1, R2 and the central bold vertex). The most striking differences are the following. The ORIG solution (top figure) finds it profitable to perform a single tour, using a single vehicle. The availability of free initial charge (ST and MT, mid and bottom figures) makes it more appealing to use both available vehicles. In the MT case (bottom figure), the technologies available in the central recharge station are different than those in stations R1 and R2. In this case, even detours might be optimal: vehicle two finds it profitable to visit customer 3 and then come back to the central recharge station, where a cheaper technology is available, before visiting customer 4, in order to avoid a more expensive recharge in station R2.

5.2.2 Capacity Cuts and Lower Bounds

As observed in the previous sections, the main drawback of a path-based formulation is the potential weakness of its lower bound. Since the gap with a route-based formulation can be reduced by means of additional inequalities, we were particularly interested in assessing the effectiveness of capacity cuts to strengthen the formulation of the master problem. For this purpose, as a first experiment, we observed the number of iterations and the computing time needed to achieve a valid dual bound at the root node with and without capacity cuts. In Table 2 we report aggregated results over groups of homogeneous instances, whose details are given in the first two columns. The number of instances in each group is reported in the third column. The table is composed by two blocks, corresponding to runs without capacity cuts and with capacity cuts. For each instance and for both techniques, the tables report the average gap between the dual bounds (DB) and the best known upper bound (BK), which (except for very few cases) are proven optimal solutions obtained by letting our branch-and-price-and-cut algorithm to run without time limits. Each gap is measured as $(BK - DB)/BK$. We also report the number of column generation iterations and the computing time needed to reach convergence (including both pricing and master problem optimization). Additionally, for the version using capacity

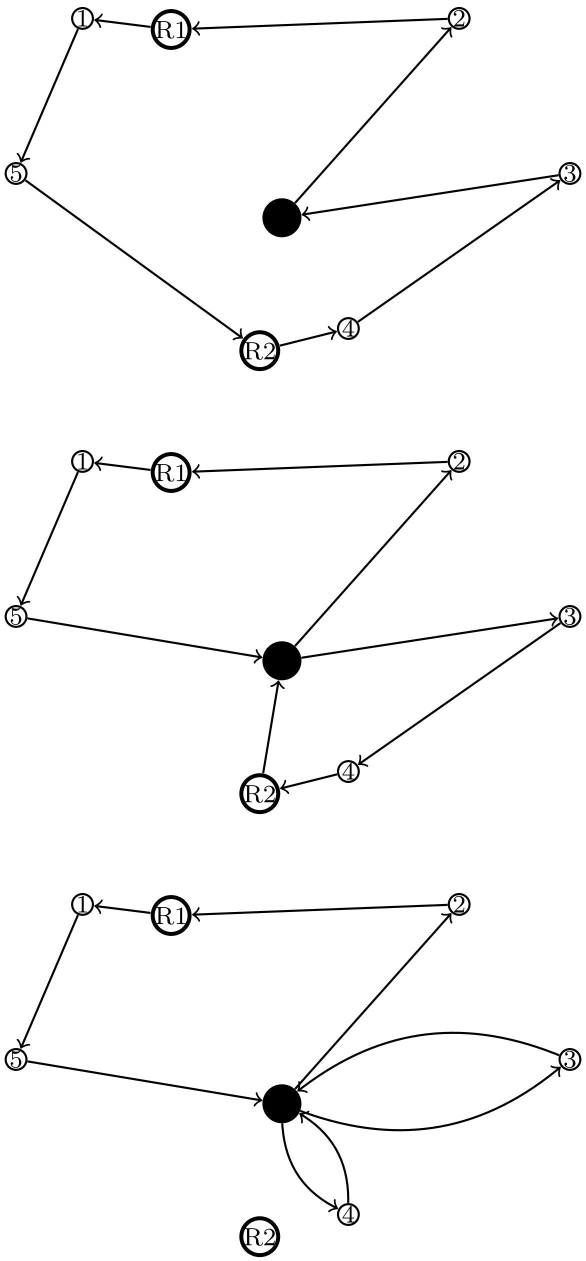

Fig. 2 Optimal solutions of instance C101-5 in three cases: ORIG (without free initial charge at the depot) - top; ST (with free initial charge at the depot and a single recharge technology at the other stations) - mid; MT (with free initial charge at the depot and multiple technologies) - bottom. Vertices 1-5 are customers, vertices R1 and R2 are recharge stations, the central black vertex represents both a recharge station and the depot

cuts we report the number of cut-and-price loops and the number of capacity cuts generated.

We noted that in all datasets, both with and without capacity cuts, a very large share of computing time was spent in pricing. The only noteworthy exception is instance A-R102-15, which is also the hardest one in dataset A. We also

Table 2 Column generation dual bounds and cutting strategies: average results

Instances			without capacity cuts			with capacity cuts				
D.S.	N	n.in.	gap	n. iter.	time	gap	n. iter.	time	rounds	cuts
A	5	12	17.1%	21.6	0.1	3.0%	28.0	0.1	10.7	15.0
	10	12	13.5%	54.9	0.1	5.4%	63.3	0.2	14.8	35.8
	15	12	11.8%	87.2	1.2	5.2%	114.8	1.9	21.5	85.8
B	30	10	6.8%	359.9	141.5	6.5%	379.1	115.8	19.3	82.4
C	10	20	15.8%	150.4	40.6	10.0%	312.2	41.9	72.4	194.9
Overall		66	13.5%	129.9	34.0	6.5%	189.5	30.6	33.4	96.4

observed that A-R102-15 required a very high number of energy consumption constraints to be separated and applied: such a phenomenon may create intricate dual structures that in turn make pricing more difficult. The number of column generation iterations needed to reach convergence tends to grow mildly as the size of the instance increases. The same applies to pricing time, excluding a few substantially harder instances (A-R102-15, B-C5-N030).

It can be observed that the effect of capacity cuts is almost negligible for the instances of dataset B. We argue that this effect is due to the clustered structure of the customers, which allows for fewer feasible combinations of paths: capacity cuts are more likely to be violated when the optimal solution of the master problem contains convex combinations of columns corresponding to paths which visit diverse set of customers, producing a sparse fractional solution. On the contrary, in clustered instances like the ones in dataset B it is less likely that the optimal solution of the master problem contains convex combinations of paths visiting customers in different clusters.

Overall, capacity cuts prove to be successful in tightening the lower bound. Indeed, such a tightening comes at the cost of a much larger number of column generation iterations needed to reach convergence. However, this has no significant impact on the average computing time at the root node. Our guess to explain this phenomenon is that capacity cuts help in rebalancing the partial dual solutions and consequently the complexity of pricing subproblem instances. In turn, more balanced pricing instances allow for more effective parallel runs, since the time needed by each single column generation iteration is often determined by the slowest among the pricing instances solved in parallel. Indeed, such a "bottleneck" effect has already been reported in the literature [4]. We have also observed that, in a few instances, adding capacity cuts reduces the number of generic cuts produced by SCIP, thereby further speeding up the resolution process.

Relying upon the results of these preliminary tests, we kept the capacity cuts active in the subsequent experiments.

5.2.3 Primal Bounds

We compared the heuristic results obtained by (a) running the column generation algorithm at the root node, and keeping the best integer solution found by the generic heuristics implemented in SCIP during column generation and (b) using the set of columns belonging to the restricted master problem in the last column generation round at the root node to build a MIP and then running CPLEX for optimizing it. In the remainder, we refer to the former procedure as *CG-based heuristics* (CGH) and to the latter procedure as *MIP-based heuristics* (MIPH). As a term of comparison, we consider the best upper bound found by the local search algorithm 48A, an ad-hoc meta-heuristics described in [13].

In Table 3, we report aggregated results over groups of homogeneous instances, whose details are given in the first five columns. The number of instances in each group is reported in the fifth column. In the subsequent columns, we include the average optimality gap of 48A. Then, for both CGH and MIPH, we report the number of instances in which the heuristics found a feasible solution, the average optimality gap on these instances and the average computing time over the whole group. The optimality gap is measured again with respect to the best known upper bound (*BK*), which (except very few cases) is given by proven optimal solutions obtained by letting our branch-and-price-and-cut algorithm run without time limits. The computing time of MIPH refers to the final MIP optimization step only and is therefore additional to that of CGH.

It is worth noting that CGH and MIPH have different nature with respect to 48A: the latter is a local search heuristic, developed ad hoc for the EVRP and therefore it exploits the combinatorial structure of the problem. On the contrary, CGH and MIPH only rely on general-purpose rounding procedures starting from the master problem fractional solution.

As a general assessment, CGH shows poor performances: in the vast majority of the cases, it could not find any feasible solution. On the contrary, when MIPH is run, a feasible solution could be found on all instances but four, that are very tightly constrained. Furthermore, on dataset A and C, the optimality gap of MIPH was consistently lower than that of 48A, remaining competitive also on the instances of dataset B, which are larger. A notable exception is instance A-RC-102-15, in which MIPH

Table 3 Comparison of primal bounds

Instances			48A	CGH			MIPH		
Dataset	N	n.inst.	gap	feas.	gap	time	feas.	gap	time
A	5	12	9.23%	8	13.05%	0.05	11	0.25%	0.03
	10	12	9.95%	1	0.00%	0.15	11	2.03%	0.10
	15	12	13.31%	4	0.89%	1.85	12	5.72%	1.09
B	30	10	0.12%	1	0.13%	115.75	9	3.18%	20.85
C	10	20	1.12%	0	’-	41.93	19	0.10%	24.91
Overall		66	6.27%	14	7.72%	30.62	62	2.00%	10.93

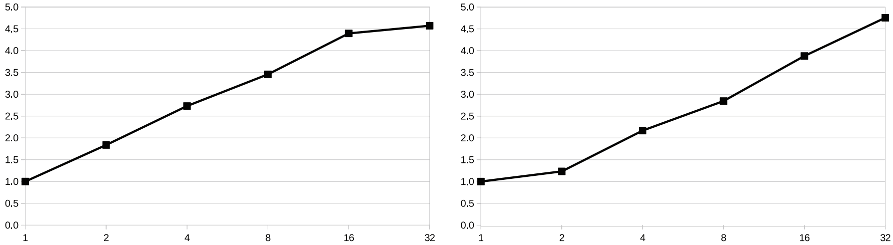

Fig. 3 Speedup factor as the number of available cores increases on dataset B (left) and dataset C (right)

leaves a large gap (while 48A does not). In terms of computing time, it was always possible to achieve both CGH and MIPH convergence within very few minutes, except for B-C5-N030 for which the column generation process required about 16 minutes of computation.

5.2.4 Parallel Pricing

One of the promising features of path-based formulations is the possibility of speeding up the pricing procedures through parallelization; this is indeed one of the main features of our algorithm.

We evaluated the scalability of our method as the amount of available computing resources, and in particular the number of CPU cores increases. Our PC is equipped with a CPU composed by 16 cores, whose architecture allows the management of up to 32 hardware threads. Therefore, we ran each column generation procedure six times, one for each value of $p \in \{1, 2, 4, 8, 16, 32\}$. At each run we restricted the process to use only p hardware threads. The number of logical threads, instead, was kept constant over the experiment.

The results of this experiment are shown in Fig. 3 and Fig. 4. Results on dataset A are omitted, since the column generation procedure was too quick to provide meaningful insights. In detail, Fig. 3 plots the speed-up factor (y axis) as the number of hardware threads (x axis) increases. For each instance, the speed-up factor obtained in a run with p hardware threads is defined as the ratio between the computing time of that run and the computing time of the run with only one hardware thread enabled. Thus, the higher the better, utopia parallelization leading to linear

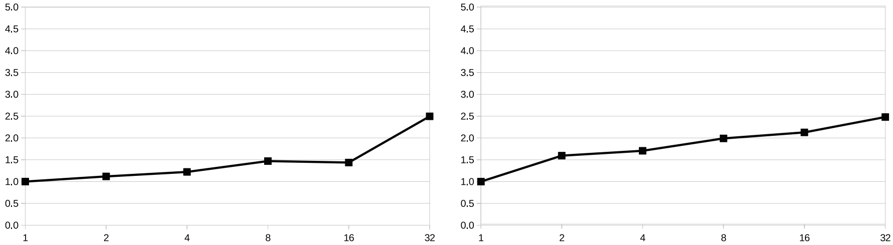

Fig. 4 Relative overall CPU time as the number of available cores increases on dataset B (left) and dataset C (right)

speed-up. Figure 3 reports average values over all the instances of dataset B (left) and dataset C (right).

Figure 4, instead, plots the relative CPU time spent by the process (y axis) as the number of hardware threads (x axis) increases. The relative CPU time in a run with p hardware threads is defined as the ratio between the overall CPU time spent on that run and the overall CPU time of the run with only one hardware thread enabled. Intuitively, such a ratio indicates the additional CPU effort for running many threads in parallel: the lower the better, utopia parallelization leading to unitary (constant) relative effort. In our case, the overhead includes the CPU time spent in solving pricing sub-problem instances that could be saved in a sequential implementation. When up to H instances of the pricing sub-problem are solved in parallel, the computing time taken by a block is due to the instance that requires the longest processing time although negative reduced cost columns have been found in other instances. On the contrary, in a sequential implementation it would be possible to stop solving pricing sub-problem instances as soon as a column with negative reduced cost were generated.

As above, average values over all the instances of dataset B (left) and dataset C (right) are reported. Values on the x axis are indicated in logarithmic scale.

Our parallelization scheme proved successful, achieving speed-up factors that appear logarithmic in our tests. No asymptotic bottleneck was reached using up to 32 parallel hardware threads. The relative CPU time grows less than linearly with the number of threads, and this yields a significant speed-up.

5.2.5 Branch-and-Cut-and-Price

Finally, we assessed the performance of the overall BCP algorithm. We observed the overall computing time, the number of sub-problems generated, the final upper and lower bounds and the corresponding gap. We stopped the BCP algorithm when the gap was reduced to 0.1%, since this is the numerical precision used by the LP optimizer. We also set a computing time limit of 3 hours. No "out of memory" condition was observed.

Average results are reported in Table 4, whose structure is similar to the previous tables: besides the characteristics of each instance group, we report the number of

Table 4 Branch-and-cut-and-price: average results

Instances			exact optimization				
Dataset	N	n. inst.	feas.	opt.	gap	nodes	time
A	5	12	12	12	0.00%	75.00	0.10
	10	12	11	11	0.00%	405.45	2.20
	15	12	12	12	0.00%	48,822.33	1,067.19
B	30	10	8	8	0.00%	2,528.10	1,210.94
C	10	20	19	14	13.41%	4,625.19	5,941.50
Overall		66	62	57	0.22%	11,319.87	1,854.39

instances for which a feasible solution was found, the number of instances solved to proven optimality, the average gap between primal bound (PB) and dual bound (DB) of the instances for which a feasible solution was found but optimality was not proven within the time limit, expressed as $(PB - DB)/DB$, the average number of branch-and-bound nodes and the average computing time for the instances solved to proven optimality.

All instances in dataset A were solved, except one (A-RC102-10) for which no feasible solution was found within the time limit. However, two of them (A-R105-15, A-RC103-15) required many nodes to be explored. Two among the ten largest instances in dataset B could not be closed within the time limit. Also in this case, no feasible solution was found within the time limit.

Several observations arise from Table 4. The performances on dataset B show remarkable differences with respect to the results on dataset A and C. First of all, some of the instances in dataset B are very tightly constrained: they allow for very few feasible solutions. This explains why the branch-and-cut-and-price algorithm failed to find feasible solutions in two cases out of ten. The second observation concerns the role of clustered customers: this feature helps when a path-based formulation is used, because the main drawback of a path-based formulation is to allow for fractional combination of paths that do not correspond to feasible routes, but this is less likely to happen when the paths generated by the pricing algorithm tend to include customers in the same cluster. This explains the good performance of the branch-and-cut-and-price algorithm, shown by the primal-dual gap achieved in dataset B: the effect of customer clusters is stronger than that related to the size of the instances. For what concerns computing time, the overall size of the instance is not so relevant on the pricing with a path-based formulation, because starting and ending stations are fixed, and the only customers which are meaningful to be included in the path are those of clusters close to the stations. Therefore it is not surprising that larger clustered instances (dataset B) could be solved easier than smaller non-clustered instances (datasets A and C).

At the same time, by comparing the results of dataset B with those of dataset A, it is clear that the branch-and-cut-and-price scales well when more than a single technology is available in each station.

6 Conclusions

The EVRP with multiple technologies we have addressed proves to be definitely more challenging than the versions of the EVRP already studied in the literature: the presence of continuous variables and the resulting mixed-integer model make the problem much more difficult than classical EVRP on graphs of the same size.

The path-based formulation we have investigated was expected to provide weak lower bounds with respect to a more common route-based formulation. However, against intuition, the main issue with our path-based formulation is not the weakness of the dual bounds: they can be effectively tightened with a limited amount of additional capacity cuts. Instead, symmetry is an issue, making the design of effective branching rules very hard. Primal feasibility is also an issue: several times the first feasible primal solution was found only after several branching operations. However, this may be due to the limited battery capacity or to other characteristics of the datasets, since some instances in our dataset are very tightly constrained. Our experiments confirm that the effect of customer clusters can be stronger than that related to the size of the instances.

The true advantage of the path-based formulation emerges from a parallel implementation of the pricing step, since the path-based formulation allows up to $KR(R-1)$ occurrences of the pricing algorithm to be executed in parallel, being K the number of vehicles and R the number of stations. Parallelization is scalable: no asymptotic bottleneck was encountered when up to 32 hardware threads were executed.

Our EVRP is amenable of diverse formulation options. For instance, modeling techniques for limiting the dependency from vehicle indices in the master, thus reducing both the model size and the risk of symmetries, appear promising. In fact, in our model there can be identical paths encoded in multiple path variables only to be available for different vehicles. However, we report that our preliminary investigations in that direction did not pay off from a computational point of view: our parallel multiple pricing strategy proved to be more effective than techniques relying on the parsimony of column variables. The search for alternative models on EVRPs is certainly an interesting and challenging research topic.

The branch-and-cut-and-price algorithm we have developed and tested also provides a very good starting point for the development of heuristics: MIPH, in spite of being based on a general-purpose solver, outperformed the specialized local search heuristic 48A in terms of solution quality, still requiring limited computing time.

We conclude with an afterthought. Our research on the EVRP is a remarkable example on how the availability itself of well understood decomposition methods and column generation algorithms was a fundamental motivating factor for the investigation of structural EVRP properties (in our case, the combinatorial properties of paths and their connections). We believe this phenomenon to be common among researchers in our area. That is, such availability *stimulates* scholars to *think* in terms of decompositions before using decomposition methods to design algorithms. Even if it sometimes fades into the background of remarkable computational results, it may arguably be the most significant heritage of seminal works as [29]. Yet, our use of parallelization for pricing (and its convincing experimental validation) is a clear example on how massive problem decomposition has only begun to show its potential. That is, massive decomposition proves to be a powerful tool for the design of algorithms which need to be scalable on next-generation architectures, having far more (less energy intensive) CPU cores.

Appendix: Details of Datasets

In Tables 5, 6 and 7, we report the details of the instances included in our datasets. The structure of the tables is similar: they contain one row for each instance, and one column for each model parameter. Instance names are given in the first column; parameter names are given in the first row.

Table 5 Details of dataset A [24]. $|\mathcal{H}| = 1$ for all instances

Instance	$\|\mathcal{N}\|$	Original K	Modified K	$\|\mathcal{R}_0\|$
A-C103-5	5	1	1	2
A-C206-5	5	1	1	4
A-C208-5	5	1	1	3
A-R202-5	5	1	1	3
A-R203-5	5	1	1	4
A-RC204-5	5	1	1	4
A-RC208-5	5	1	1	3
A-C101-5	5	2	2	3
A-R104-5	5	2	2	3
A-R105-5	5	2	2	3
A-RC105-5	5	2	2	4
A-RC108-5	5	1	2	4
A-C202-10	10	1	1	5
A-R201-10	10	1	1	4
A-R203-10	10	1	1	5
A-RC201-10	10	1	1	4
A-C101-10	10	3	2	5
A-C104-10	10	2	2	4
A-C205-10	10	2	2	3
A-R103-10	10	2	2	3
A-RC205-10	10	2	2	4
A-R102-10	10	3	3	4
A-RC102-10	10	4	3	4
A-RC108-10	10	3	3	4
A-R209-15	15	1	1	5
A-RC204-15	15	1	1	7
A-C103-15	15	3	2	5
A-C202-15	15	2	2	5
A-C208-15	15	2	2	4
A-R202-15	15	2	2	6
A-RC108-15	15	3	2	5
A-RC202-15	15	2	2	5
A-C106-15	15	3	3	3
A-R105-15	15	4	3	6
A-RC103-15	15	4	3	5
A-R102-15	15	5	4	8

Table 6 Details of dataset B

Instance	$\vert\mathcal{N}\vert$	K	$\vert\mathcal{R}_0\vert$	$\vert\mathcal{H}\vert$
B-C0-N030	30	7	5	3
B-C1-N030	30	7	5	3
B-C2-N030	30	7	5	3
B-C3-N030	30	7	5	3
B-C4-N030	30	7	5	3
B-C5-N030	30	7	5	3
B-C6-N030	30	7	5	3
B-C7-N030	30	7	5	3
B-C8-N030	30	7	5	3
B-C9-N030	30	6	5	3

Table 7 Details of dataset C [13]

Instance	$\vert\mathcal{N}\vert$	K	$\vert\mathcal{R}_0\vert$	$\vert\mathcal{H}\vert$
C-10-N10	10	4	9	3
C-11-N10	10	4	9	3
C-12-N10	10	4	9	3
C-13-N10	10	4	9	3
C-14-N10	10	5	9	3
C-15-N10	10	4	9	3
C-16-N10	10	4	9	3
C-17-N10	10	5	9	3
C-18-N10	10	5	9	3
C-19-N10	10	4	9	3
C-20-N10	10	4	5	3
C-21-N10	10	4	5	3
C-22-N10	10	4	5	3
C-23-N10	10	3	5	3
C-24-N10	10	4	5	3
C-25-N10	10	3	5	3
C-26-N10	10	4	5	3
C-27-N10	10	4	5	3
C-28-N10	10	4	5	3
C-29-N10	10	4	5	3

Acknowledgements The fourth author acknowledges the support of the programme “Visitantes Distinguidos 2011-12” of Universidad Complutense de Madrid. The third and fifth authors acknowledge the support of the Government of Madrid, project S2013/ICE-2845, and the Government of Spain, project MTM2015-65803-R. The authors are grateful to the participants of the Odysseus 2015 conference for their useful comments on a preliminary version of this work, and to two anonymous referees, whose insights helped to substantially improve the paper. Dario Bezzi and Saverio Basso deserve special thanks for providing a useful tool to produce figures and for several insights on parallelization techniques and technologies, respectively.

Funding Open Access funding provided by Università degli Studi di Milano

Declarations

Conflicts of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Andelmin J, Bartolini E (2017) An exact algorithm for the green vehicle routing problem. Transportation Science 51(4):1288–1303
2. Barnhart C, Boland NL, Clarke LW, Johnson EL, Nemhauser GL, Shenoi RG (1998) Flight String Models for Aircraft Fleeting and Routing. Transportation Science 32(3):208–220
3. Baldacci R, Mingozzi A, Roberti R (2011) New route relaxation and pricing strategies for the vehicle routing problem. Operations Research 59:1269–1283
4. Basso S, Ceselli A (2017) Asynchronous Column Generation, Proc. of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX)
5. Ceselli A, Righini G (2020) The Electric Traveling Salesman Problem: properties and models, Technical Report 2434/789142 - University of Milan https://doi.org/10.13140/RG.2.2.17712.99848
6. Breunig U, Baldacci R, Hartl RF, Vidal T (2018) The Electric Two-Echelon Vehicle Routing Problem, Technical Report. Available at: https://arxiv.org/pdf/1803.03628.pdf
7. Bruglieri M, Mancini S, Pisacane O (2018) Solving the green vehicle routing problem with capacitated alternative fuel stations, in Proceedings of 16th Cologne-Twente Workshop on Graphs and Combinatorial Optimization. France, Paris, pp 196–199
8. Christofides N, Mingozzi A, Toth P (1981) Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations. Mathematical Programming 20:255–282
9. Conrad RG, Figliozzi MA (2011) The recharging vehicle routing problem, Proceedings of the 2011 Industrial Engineering Research Conference, T. Doolen, E. van Aken eds., Portland, USA
10. Desaulniers G, Errico F, Irnich S, Schneider M (2016) Exact algorithms for electric vehicle-routing problems with time windows. Opererations Research 64(6):1388–1405
11. Erdogan S, Miller-Hooks E (2012) A green vehicle routing problem. Transportation Research Part E 48:100–114
12. Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. Networks 44:216–229
13. Felipe Ortega A, Ortuño Sánchez MT, Righini G, Tirado Domínguez G (2014) A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges, Transportation Research Part E, 71:111-128
14. Hiermann G, Puchinger J, Ropke S, Hartl RF (2016) The electric fleet size and mix vehicle routing problem with time windows and recharging stations. Eur J Oper Res 252(3):995–1018
15. Keskin M, Çatay B (2018) A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. Comput Oper Res 100:172–188
16. Keskin M, Laporte G, Çatay B (2019) Electric Vehicle Routing Problem with Time-Dependent Waiting Times at Recharging Stations. Comput Oper Res 107:77–94
17. Koç C, Jabali O, Laporte G (2018) uthors Jacques Desrosiers. Long-haul vehicle routing and scheduling with idling options, Journal of the Operations Research Society 69(2):235–246
18. Koyuncu I, Yavuz M (2019) Duplicating nodes or arcs in green vehicle routing: A computational comparison of two formulations. Transportation Research Part E 122:605–623

19. Li-Ying W, Yuan-Bin S (2015) Multiple charging station location-routing problem with time window of electric vehicle. J Eng Sci Technol Rev 8(5):190–201
20. Lysgaard J, Letchford AN, Eglese RW (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming 100:423–445
21. Montoya A, Guaret C, Mendoza JE, Villegas JG (2017) The electric vehicle routing problem with nonlinear charging function, Transportation Research B 103:87-110
22. Pelletier S, Jabali O, Laporte G (2016) Goods distribution with electric vehicles: review and research perspectives. Transportation Science 50(1):3–22
23. Sassi O, Cherif WR, Oulamara A (2014) Vehicle Routing Problem with Mixed Fleet of Conventional and Heterogenous Electric Vehicles and Time Dependent Charging Costs, Technical Report. Available at: https://hal.archives-ouvertes.fr/hal-01083966/
24. Schneider M, Stenger A, Goeke D (2014) The electric vehicle-routing problem with time windows and recharging stations. Transportation Science 48:500–520
25. Schneider M (2014) Personal Communication
26. SCIP: Solving Constraint Integer Programs, scip.zib.de, last accessed 7.4.2015
27. Sweda TM, Dolinskaya IS, Klabjan D (2017) Adaptive routing and recharging policies for electric vehicles. Transp. Sci. 51(4):1326–1348
28. Villegas J, Guaret C, Mendoza JE, Montoya A (2018) The Technician Routing and Scheduling Problem with Conventional and Electric Vehicle, Technical Report. Available at: https://hal.archives-ouvertes.fr/hal-01813887/document
29. Desrosiers J, Soumis F, Desrochers M (1984) Routing with time windows by column generation. Networks 14(4):545–565

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Alberto Ceselli[1] · Ángel Felipe[2] · M. Teresa Ortuño[2] · Giovanni Righini[1] · Gregorio Tirado[3]

Ángel Felipe
felipe@mat.ucm.es

M. Teresa Ortuño
tortuno@mat.ucm.es

Giovanni Righini
giovanni.righini@unimi.it

Gregorio Tirado
gregoriotd@ucm.es

[1] Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, 20133 Milano, Italy

[2] Departamento de Estadística e Investigación Operativa I, Universidad Complutense de Madrid, Plaza de Ciencias 3, 28040 Madrid, Spain

[3] Departamento de Estadística e Investigación Operativa II, Universidad Complutense de Madrid, Campus de Somosaguas, 28223 Pozuelo de Alarcón, Spain