Arc Routing with Electric Vehicles

Elena Fernández[1], Markus Leitner[2], Ivana Ljubić[3] and Mario Ruthmair[4]

[1]Department of Statistics and Operations Research, University of Cádiz, Spain.
elena.fernandez@uca.es
[2]Department of Supply Chain Analytics, Vrije Universiteit Amsterdam, The Netherlands.
m.leitner@vu.nl
[3]ESSEC Business School of Paris, Cergy-Pontoise, France. ivana.ljubic@essec.edu
[4]University of Vienna, Department of Statistics and Operations Research, Vienna, Austria.
mario.ruthmair@univie.ac.at

June 21, 2019

Abstract

Concerns about greenhouse gas emissions and government regulations foster the use of electric vehicles. Several recently published articles study the use of electric vehicles (EVs) in node-routing problems. In contrast, this article considers EVs in the context of arc routing while also addressing practically relevant aspects that have not been addressed sufficiently so far. These include dynamic charging of EVs while driving, speed-dependent energy consumption, and non-linear charging functions that depend on the battery state-of-charge and the charging time. A generic way of dealing with these aspects is introduced through the concept of an energy-indexed graph which is used to derive an integer linear programming formulation and an exact solution framework based on branch-and-cut. Efficient construction heuristics and a local search for approximately solving large-scale instances are proposed. These heuristics build upon two different solution encodings for which labeling algorithms for decoding are introduced. A computational study is performed on realistic problem instances. Besides analyzing the performance of all proposed methods, the obtained results provide insights and recommendations on strategic decisions related to the size of the vehicle fleet, the battery size, and the amount of charging facilities.

Keywords: arc routing, electric vehicles, speed-dependent energy consumption, non-linear charging, branch-and-cut, heuristics

1 Introduction

Increasing driving ranges together with low maintenance costs foster the replacement of vehicles with combustion engines by (battery) electric vehicles (EVs) [34]. This trend also holds for company fleets in which vehicles may be (partly) replaced by EVs that can be conveniently recharged (overnight) using charging stations that are installed at car parks of the respective companies. Nevertheless, the use of EVs imposes additional challenges (compared to combustion engine vehicles) since time-demanding charging breaks during service may be necessary in the case of long trips and since their energy consumption heavily depends on the driving speed (among other factors) [2, 35].

The worldwide fleet of EVs grew 54% to about 3.1 million in 2017, and according to the International Energy Agency, this number is forecasted to hit 125 million by 2030 [23]. Nevertheless, EVs still make up less than 1% of passenger vehicles worldwide. According to [27], one of the major pain points for a wide adoption of EVs is the sparse or non-existing infrastructure. Thus, drivers need to identify appropriate charging stations before trips and reserve time to charge up the batteries. Classical *stationary charging* in

garages and parking lots, and *opportunity charging* (at, e.g., bus stops or shopping malls), require the drivers to include charging times in their itineraries and do not fully remove the range anxiety, one of the major barriers for adopting EVs at a large scale. *Dynamic charging* techniques that allow to recharge a vehicle while being driven can play a significant role in the massive adoption of electric vehicles, both for passenger and cargo transportation. This is because dynamic charging reduces the need for overnight or stationary charging and increases the reliability of EVs. In the long run it also allows for lowering the price of EVs, due to the fact that smaller batteries would be sufficient, as they only need to supply power between segments with dynamic charging infrastructure.

Dynamic charging can, e.g., be implemented via the inductive power transfer (IPT) technology which enables dynamic *wireless* charging along the roads [29]. In this scenario, an EV is recharged while moving along a dedicated lane equipped with an IPT system. A recent survey regarding wireless power transfer technologies for EVs can be found in [8]. An alternative technology transfers the energy from the metal rails installed in the road via a movable arm attached to the bottom of an electric vehicle, or a truck. This technology is implemented on street segments at the Stockholm airport Arlanda in the pioneering project *eRoadAlanda* [1].

Overview and scientific contribution. This article provides a first and comprehensive study on the use of EVs in the context of arc routing with possible dynamic charging. Given a (street) network including a set of required arcs, the *Electric Arc Routing Problem (eARP)* proposed in this article is to find a set of routes that visit all required arcs with minimal total travel time while respecting the energy usage constraints imposed by the EVs. Charging of EVs is possible along arcs (dynamic charging) during service and at the depot node (stationary/opportunistic charging) before and after service. Though stationary charging could be easily integrated in our problem formulations and solution approaches, we focus on dynamic charging that has the benefit of avoiding time-consuming recharging breaks during service. The main contributions of this paper can be summarized as follows:

- We introduce the concept of the *energy-indexed graph* that provides a generic way to deal with (i) the possibility of dynamic charging, (ii) speed dependent energy consumption, and (iii) non-linear charging functions that depend on the initial battery state-of-charge (SOC) and the charging time. To the best of our knowledge, the first two aspects have not been considered in the context of EVs yet, while the third one has been recently addressed for node-routing problems by Montoya et al. [32].

- We derive a mathematical model on the energy-indexed graph and develop an exact solution framework based on branch-and-cut which provides a useful decision support tool for routing a homogeneous fleet of vehicles so that all arcs can be served within the shortest total travel time, without exceeding the battery capacity.

- We propose two different ways for encoding eARP solutions, we study the complexity of the underlying feasibility problem of a given encoding, and propose labeling algorithms for their decodings.

- Based on the latter result, we also derive efficient heuristics for obtaining feasible routes. The heuristics can be used as a stand-alone approach, or for the initialization of the exact solution framework.

- We provide an analytical description of a piece-wise linear approximation of the SOC function for dynamic charging.

- Finally, to provide managerial insights, we conduct a computational study on realistic problem instances and analyze in detail the obtained optimal solutions for different battery sizes and different distributions of dynamic charging arcs. We give insights and recommendations on strategic decisions related to the size of the vehicle fleet, the battery size, and the amount of charging facilities.

The paper is organized as follows: In the remainder of this section we provide a literature overview and a formal problem definition. The energy-indexed graph is presented in Section 2 along with the Integer Linear Programming (ILP) formulation. In Section 3 we discuss two different solution encodings and the

corresponding decoding procedures. Section 4 studies heuristics, whereas the algorithmic details of our branch-and-cut implementations are given in Section 5. A computational study and managerial insights are given in Section 6, and final conclusions are drawn in Section 7.

1.1 Related work

The use of electric vehicles has raised multiple issues in application fields related to transportation logistics, green logistics and vehicle routing in general. A recent excellent survey of the existing research in the field is given by Pelletier et al. [34]. Most of the optimization problems resulting from transportation models that integrate the use of EVs are extensions of vehicle routing problems (VRPs) dealing with the limited autonomy of EVs, time-dependency issues and, to a very limited extent battery degradation, see Pelletier et al. [35]. Given the difficulty of the resulting models, they are typically addressed with approximate heuristic methods.

Several papers study the relation of energy consumption with the velocity of vehicles. Bektas and Laporte [6] propose the Pollution-Routing Problem in which the operational cost, the number of drivers and the cost of the green house emissions are simultaneously optimized by controlling the major factors that affect the emissions, namely the vehicle speed and the load. To derive a tractable ILP model, the authors discretize the speed function by using a small set of speed levels. In a follow-up work of Franceschetti et al. [18], the authors propose the Time-Dependent Pollution-Routing Problem where the routes for a fleet of vehicles that serve a set of customers must be determined together with the speeds on each leg of the routes. The cost function takes into account traffic congestion which, at peak periods, significantly restricts vehicle speeds and increases emissions. Fukasawa et al. [19] also combine route and speed optimization for VRPs, assuming that the fuel cost function is a strictly convex differentiable function of the average travel speed over an edge. A branch-and-price algorithm is proposed with a tailored labeling algorithm to deal with the non-linear pricing function. In a related work of Qian and Eglese [36], the authors model the traffic congestion by assuming that the speed along each edge depends on the time of the day. The authors consider a discretized time-horizon in which a VRP is solved so that the overall CO2 emissions are minimized, while the time consumed for each route is bounded by a constant.

More recently, Ferro et al. [16] study an extension of the Green VRP, which adds time-variant prices for energy purchase, and different EV consumption and charging modes. The main decisions refer to the velocity of EVs, the loaded cargo and the battery charge at recharging nodes. The objective is the minimization of the cost for the total travel distance and that for energy purchase depending on the selected recharging modes. Recently, Macrina et al. [31] introduce a new variant of the Green VRP with time windows where traditional and electrical vehicles jointly operate, and the limited autonomy of the batteries of electric vehicles is taken into account. The possibility of recharging partially the batteries at any of the available stations is considered, together with a limitation on the polluting emissions for the conventional vehicles. The behavior of the proposed approach is evaluated empirically on a large set of test instances.

While many authors have studied transportation problems incorporating the implications derived from the use of electric vehicles within distribution management, very few papers deal with the effect of battery degradation. Most existing electric VRP models assume that the battery-charge level is a linear function of the charging time [14, 39] although in reality the function is non-linear, and difficult to integrate within mathematical programming models, as its evaluation involves solving differential equations. Pelletier et al. [35] address this issue and discuss tractable models for transportation problems that will allow estimate charging and discharging behavior of EV batteries. Montoya et al. [32] and Zündorf [43] consider electric vehicle routing with realistic (non-linear) charging functions and several approximations for them. Charging functions are defined as two-dimensional functions depending on SOC and time where only charging at nodes is possible. The authors assume concave charging functions since no (relevant amount of) energy is consumed during charging at a station (which is different for dynamic charging while driving).

In the eARP service demand is placed at the arcs of a directed network. Thus, from the perspective of the design of vehicle routes the eARP closely relates to ARPs [9, 13]. In particular, the eARP is equivalent to the well-known *Directed Rural Postman problem (DRPP)* [33] in case the battery capacity is not restrictive. In Section 2 we will give an alternative definition of the eARP that also relates it to the *generalized directed*

rural postman problem (GDRPP) introduced by Ávila et al. [3]. Aspects derived from the incorporation of wireless technologies within ARPs have been studied in the close-enough ARP [22, 40] that is closely related to the GDRPP. In the close-enough ARP vehicles must traverse a given set of arcs to measure customers consumption, and a reading device is installed in the vehicles to collect the data sent by the metering devices. However, the close-enough ARP considers the use of conventional vehicles and wireless technology is used for capturing the metering information. There are some similarities between the eARP and the capacitated ARP with deadheading demand proposed by Bartolini et al. [4] when we interpret the demands as energy consumption and the vehicles' load capacity as battery capacity. In the eARP, however, we allow more than one traversal options, recovering of the consumed resource (battery charging), and more complicated functions to derive the resource consumption for a traversal. To the best of our knowledge there is no work in the literature where issues derived from the use of EVs are incorporated within an ARP.

1.2 Problem definition

An eARP instance is defined on a directed graph $G = (V, A)$ representing the underlying (street) network. Node set V contains a depot node 1 at which m identical, initially fully charged, EVs with a battery capacity of Q are located. Arc set A contains the set $A_{\mathrm{R}} \subseteq A$ of required arcs that need to be traversed by at least one such vehicle. A set of travel times $T(a)$ (typically) resulting from the possible speeds for traversing arc a is associated with each arc $a \in A$.

A solution to the eARP is a collection of *feasible walks along with their associated travel times* (one for each EV) that covers all required arcs. A walk $\mathcal{W} = (a^1, \ldots, a^k)$ with associated travel times $\mathcal{T} = (t^1, \ldots, t^k)$, $t^j \in T(a^j)$, $a^j \in A$, for all $1 \leq j \leq k$, is feasible if the following holds:

(i) $\mathcal{W}$ starts and ends at the depot,

(ii) The SOC b^j after traversing arc $a^j = (u^{j-1}, u^j) \in \mathcal{W}$ must be in the interval $[0, Q]$ for all $0 \leq j \leq k$.

SOC b^j after traversing the arc $a^j = (u^{j-1}, u^j) \in \mathcal{W}$ with travel time t^j is defined as

$$b^j = \beta(a^j, t^j, b^{j-1}), \quad 1 \leq j \leq k \tag{1}$$

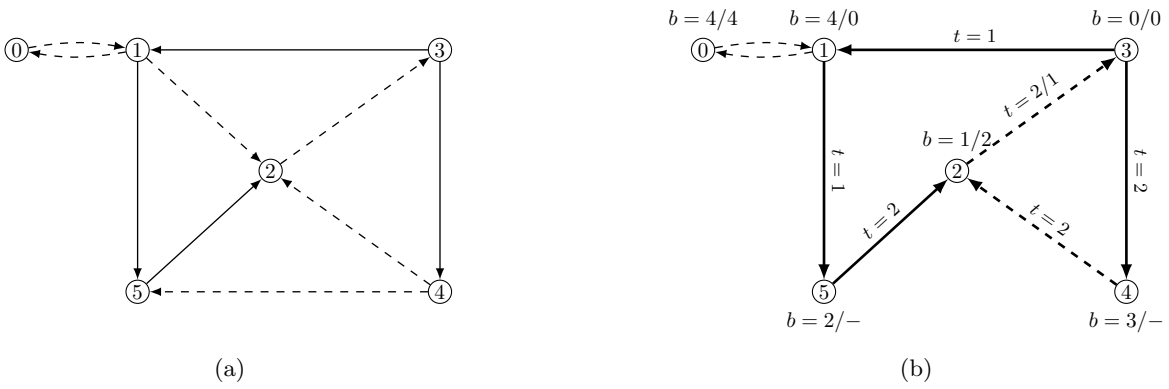
where the initial SOC at the depot is $b^0 = Q$ and β is a generic (non-linear) function describing the relation between the SOC b^{j-1} before entering arc a^j and the SOC b^j after traversing it with time t^j. The objective is to identify a set of m feasible walks $S = \{(\mathcal{W}^\ell, \mathcal{T}^\ell)\}_{\ell=1}^m$ with minimum total travel time that covers all the required arcs. As mentioned above, the eARP boils down to the DRPP in case the battery capacity is sufficiently large. Since the DRPP is known to be NP-hard [26], we conclude that the eARP is NP-hard too.

1.3 Assumptions, notation, and an example

In the remainder of this article, we will assume that the set of travel times $T(a)$ for each arc $a \in A$ is finite. Furthermore, we use B to denote the set of SOC values an initially fully charged vehicle may reach at any node $u \in V$ and assume that B is finite too. The latter assumption is without loss of generality as long the set of travel times is finite and each arc is traversed only a finite number of times.

Since each travel time results from a particular (average) driving speed along an arc, one may, however, argue that this set actually contains an infinite number of elements. To this end, we note that it is unlikely to know the precise driving speed along each arc in advance while planning the routes to be driven later on. This issue is also discussed in [35], where discretization is presented as an alternative for modeling the behavior of a battery during discharging and charging. One advantage of discretization is that it avoids dealing with complex continuous temporal functions that model the SOC, which can be very difficult to integrate within mathematical programming models as their evaluation involves solving differential equations [35].

The relations between travel time, initial and final SOC for each arc $a \in A$ and $t \in T(a)$ are described via the *battery SOC function* $\beta : A \times \mathbb{R}^+ \times B \mapsto B \cup \{-\infty\}$ where the value $\beta(a, t, b) = -\infty$ if it is infeasible to cross the arc a with travel time t and initial SOC b. We also assume that drivers behave rationally and



(a) (b)

Figure 1: (a) An example instance and (b) an optimal solution to it. Required arcs are shown by solid lines, SOC values for the first and (where applicable) second visit of each node are given next to the nodes, and used travel times are given next to the arcs.

hence do not drive unnecessarily slow in case a smaller travel time would not induce a smaller SOC after traversing an arc. This assumption implies that $\beta(a,t,b) \leq \beta(a,t',b')$ holds for each $a \in A$ and $t \leq t'$, $b \leq b'$ whenever the respective values of β are finite.

Since the depot 1 can be visited multiple times within a single walk, we also augment graph G by an artificial depot node 0 which serves as the unique start and end point of each vehicle's tour. Node 0 has incident arcs $(0,1)$ and $(1,0)$ connecting it to the real depot with an associated travel time of zero, i.e., $T((0,1)) = T((1,0)) = \{0\}$. No energy is consumed along the arc $(0,1)$, and a vehicle will be fully charged when returning to the artificial depot 0, i.e., $\beta((0,1),0,Q) = Q$ and $\beta((1,0),0,b) = Q$, for all $b \in [0,Q]$.

Figure 1a shows an example with $A_{\mathrm{R}} = \{(1,5),(3,1),(3,4),(5,2)\}$, $m = 1$, $Q = 4$, and two possible travel times $T(a) = \{1,2\}$ for each $a \in A$. We assume that the SOC function is defined as $\beta(a,1,b) = b$ and $\beta(a,2,b) = \min\{Q, b+3\}$ for charging arcs $a \in \{(3,1),(3,4)\}$ and $b \in B$. For the remaining arcs $a \in A \setminus \{(3,1),(3,4)\}$, we set $\beta(a,1,b) = b-2$, for all $b \in B$, $b \geq 2$, $\beta(a,2,b) = b-1$, for all $b \in B$, $b \geq 1$, and $\beta(a,t,b) = -\infty$ otherwise. Figure 1b visualizes an optimal solution with total travel time 11 that consists of walk $((0,1),(1,5),(5,2),(2,3),(3,4),(4,2),(2,3),(3,1),(1,0))$ where a travel time of 1 is chosen on arcs $(1,5)$, $(2,3)$ (for the second traversal), $(3,1)$ and of 2 on all other arc traversal not adjacent to the artificial depot.

2 Integer programming formulations

Observe that eARP instances may contain *positive SOC cycles* which are cycles in G whose traversal can increase a vehicle's SOC. One such example is given in Figure 1 where traversing the cycle $(2,3,4,2)$ can induce an SOC increase of one. The existence of such positive SOC cycles and multiple node/arc visits (even by a single vehicle) increase the difficulty of deriving ILP formulations on the original graph G. This is because it is not obvious how to track the battery's SOC with the usual state variables on nodes and arcs in particular for non-linear SOC functions. Formulations based on node-routing transformations are an alternative commonly applied to ARPs, see, e.g., [7]. These transformations rely on the fact that deadheading between two consecutively visited required arcs, say (u,v) and (u',v') is performed along a shortest path connecting v to u'. It is easy to observe, however, that this property does not necessarily hold for the eARP in which an optimal deadheading route has to be selected from a set of Pareto optimal walks, cf. [4] and Section 3 for further details. Hence, such a transformation would result in creating a multi-graph with a potentially exponential number of parallel edges, which might not be tractable from a computational perspective. We overcome these modeling difficulties by proposing a transformation of the eARP to an equivalent problem defined on an *energy-indexed graph* that is defined below.

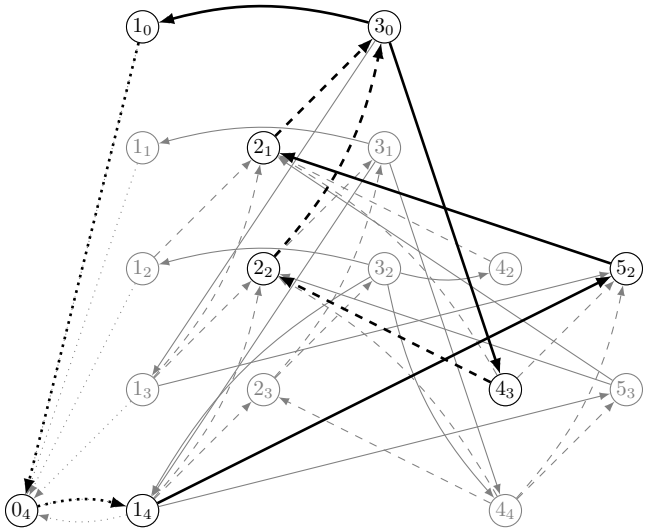


Figure 2: Energy-indexed graph corresponding to the instance given in Figure 1a. The solution given in Figure 1b is indicated by bold arcs. Artificial arcs adjacent to 0_4 are shown by dotted lines, optional arcs by dashed lines, and arcs corresponding to required arcs by solid lines. Nodes without ingoing or without outgoing arcs have been removed together with their adjacent arcs.

2.1 The Energy-Indexed Graph $\mathcal{G}$

Starting from graph G, we construct the energy-indexed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ in such a way that its nodes and arcs encode SOC information, see Definition 1 and Figure 2.

Definition 1. *Given an instance of the eARP, the energy-indexed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ associated with that instance is defined as follows:*

$$\mathcal{V} = \{u_p : u \in V \setminus \{0\}, p \in B\} \cup \{0_Q\},$$
$$\mathcal{A} = \{(u_p, v_q) : a = (u, v) \in A,\ p, q \in B,\ \textit{and } \exists t \in T(a)\ \textit{s.t. } \beta(a, t, p) = q\}.$$

To each arc $(u_p, v_q) \in \mathcal{A}$, we associate the travel time $t \in T((u, v))$, such that $\beta((u, v), t, p) = q$. By $\mathcal{A}_{uv} = \{(u_p, v_q) \in \mathcal{A}\}$ we denote the set of energy-indexed arc copies corresponding to arc $(u, v) \in A$. To simplify notation, above definition of $\mathcal{G}$ is based on introducing one copy u_p of each node u at each possible SOC $p \in B$. Observe, however, that our implementation is based on a (usually) much smaller graph that only contains nodes (and incident arcs) that are reachable via an energy-feasible walk from the artificial depot 0 (see Section 5.1 for details). It is clear that the size of $\mathcal{G}$ is notably larger than that of G both in terms of the number of nodes and the number of arcs. On the other hand, the advantage of working on $\mathcal{G}$ rather than on G is that it makes explicit the SOC at the nodes as well as the energy consumption of the traversed arcs. Moreover, as formalized in Lemma 1, by construction any walk on $\mathcal{G}$ starting and ending at the depot is battery feasible.

Lemma 1. *There is a one-to-one correspondence between the battery feasible walks on G starting and ending at depot 0 and the walks on $\mathcal{G}$ starting and ending at 0_Q.*

Proof. Let $\mathcal{W} = (a^1, a^2, \ldots, a^k)$, $a^j = (u^{j-1}, u^j) \in A$, $1 \le j \le k$, $u^0 = u^k = 0$, be a battery feasible walk in G and $\mathcal{T} = (t^1, t^2, \ldots, t^k)$, $t^j \in T(a^j)$, $1 \le j \le k$, be the associated travel times. For each j, $1 \le j < k$, subwalk $(a^1, a^2, \ldots, a^j)$ is battery feasible with a (final) SOC $q \in B$. The definition of $\mathcal{G}$ implies that $(u_p^{j-1}, u_q^j) \in \mathcal{A}$ where $q = \beta(a^j, t^j, p)$. Consequently, a walk in $\mathcal{G}$ corresponding to $\mathcal{W}$ exists. The result

follows from observing that, by construction, every walk in $\mathcal{G}$ is battery feasible and associated with its corresponding sequence of travel times. □

An immediate consequence of Lemma 1 is that the eARP can be solved by finding a set of at most m walks on $\mathcal{G}$ with minimal total travel time such that (i) each walk starts and ends at 0_Q, and (ii) at least one arc of set $\mathcal{A}_{uv}$ for each required arc $(u, v) \in A_{\mathrm{R}}$ is traversed. Thus, the eARP can be seen as a particular case of the of the *generalized directed rural postman problem (GDRPP)*, see, e.g., Ávila et al. [3], defined on the energy-indexed graph $\mathcal{G}$, with required arc subsets $\mathcal{A}_{uv}$, $(u, v) \in A_{\mathrm{R}}$, and the additional condition that the solution contains no more than m walks.

2.2 ILP formulation on the energy-indexed graph

Next, we introduce an ILP formulation, which exploits the structure of the energy-indexed graph $\mathcal{G}$ and its aforementioned relation to the eARP. The formulation considers the following sets of decision variables:

- integer variables X_a, for all $a = (u_p, v_q) \in \mathcal{A}$ indicating the number of times arc $(u, v) \in A$ is traversed with SOC level p at node u and travel time t, where $q = \beta((u, v), t, p)$; and
- binary variables y^p_{uv}, for all $(u, v) \in A_{\mathrm{R}}$, $p \in B$, that indicate if required arc (u, v) is served with SOC level p at node u.

Besides previously introduced notation, formulation (2) uses the compact notation $X(\mathcal{A}') = \sum_{a \in \mathcal{A}'} X_a$ for any set of arcs $\mathcal{A}' \subseteq \mathcal{A}$.

$$\min \quad \sum_{a \in \mathcal{A}} t_a X_a \tag{2a}$$

$$\text{s.t.} \quad X(\delta^+(0_Q)) \leq m \tag{2b}$$

$$X(\mathcal{A}_{uv}) \geq 1 \qquad \forall (u, v) \in A_{\mathrm{R}} \tag{2c}$$

$$X(\delta^-(u_p)) = X(\delta^+(u_p)) \qquad \forall u_p \in \mathcal{V} \tag{2d}$$

$$\sum_{p \in B} y^p_{uv} = 1 \qquad \forall (u, v) \in A_{\mathrm{R}} \tag{2e}$$

$$X(\mathcal{A}_{uv}) \geq y^p_{uv} \qquad \forall (u, v) \in A_{\mathrm{R}}, p \in B \tag{2f}$$

$$X(\delta^-(S)) \geq y^p_{uv} \qquad \forall (u, v) \in A_{\mathrm{R}}, S \subseteq \mathcal{V} \setminus \{0_Q\}, u_p \in S \tag{2g}$$

$$X_a \in \mathbb{Z}_+ \qquad \forall a \in \mathcal{A} \tag{2h}$$

$$y^p_{uv} \in \{0, 1\} \qquad \forall (u, v) \in A_{\mathrm{R}}, p \in B. \tag{2i}$$

Inequality (2b) ensures that no more than m walks leave the artificial depot 0_Q, while constraints (2c) guarantee that at least one arc from each set of arcs corresponding to a single required arc in G is traversed. Equations (2d) are flow conservation constraints. Assignment constraints (2e) ensure that each required arc is served exactly once. Linking inequalities (2f) ensure that if arc $(u, v) \in A_{\mathrm{R}}$ is served with an initial SOC of p then some arc $(u_p, v_q) \in \mathcal{A}$ has to be traversed. Closed subwalks that serve a required arc but are not connected to the depot are prevented by constraints (2g) together with (2d)-(2f). Whenever arc $(u, v) \in A_{\mathrm{R}}$ is served with initial SOC p, connectivity constraints (2g) ensure the existence of a walk from 0_Q to u_p which implies that the service traversal of each required arc must be reachable from the depot.

A somewhat natural question is whether one could eliminate variables y and constraints (2e)-(2g) related to them from formulation (2). To this end, we observe that such a model would allow for infeasible solutions even if additionally considering connectivity constraints on graph G. Indeed, cycles or walks that are not connected to the depot 0_Q and contain service traversals of required arcs may arise in solutions of such a model even when the corresponding subgraph is connected in graph G. Figures 3a and 3b show one such example on graphs G and $\mathcal{G}$, respectively, in which we assume that $Q = 4$, $A_{\mathrm{R}} = \{(2, 3)\}$, a unique travel time and a unique energy consumption for each arc. The latter is given as $\beta((u, v), t_{uv}, b) = b + \ell_{uv}$, where

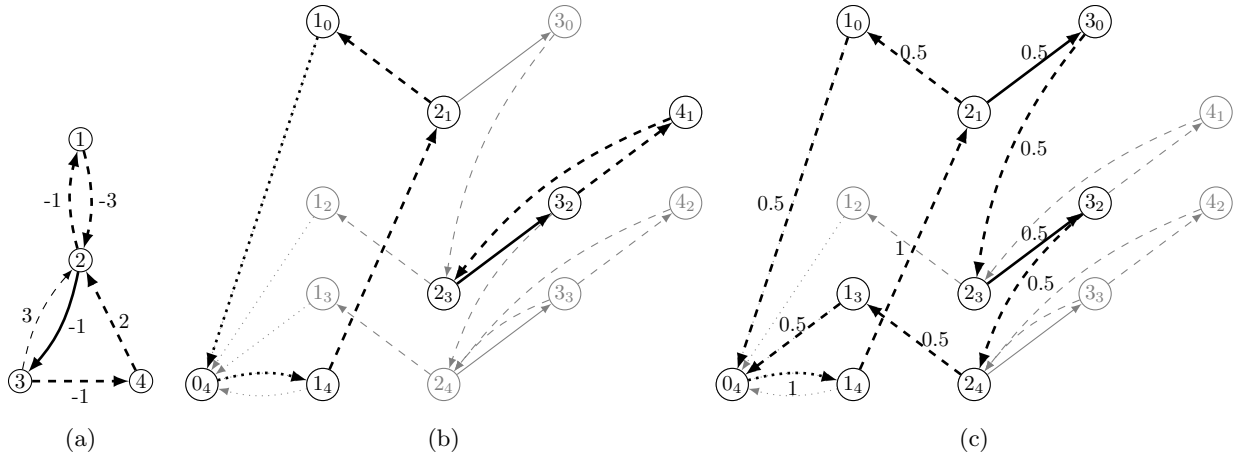


Figure 3: (a) Example eARP instance. (b) Infeasible integer solution in energy-indexed graph $\mathcal{G}$ (indicated by bold arcs) that is feasible for formulation (2) without inequalities (2e)-(2g). (c) LP solution in energy-indexed graph $\mathcal{G}$ (indicated by bold arcs with LP values) violating lifted inequalities (4). The solution given in Figure 1b is indicated by bold arcs. Artificial arcs are shown by dotted lines, optional arcs by dashed lines, and (energy-indexed copies of) required arcs by solid lines.

ℓ_{uv} is the arc label given in Figure 3a. Note that this solution would violate connectivity constraint (2g) for required arc $(2,3)$ and $S = \{2_3, 3_2, 4_1\}$, if $y^3_{23} = 1$.

Note that the feasible domain of formulation (2) contains solutions that, in addition to the walks connected to 0_Q in which the required arcs are served, include closed subwalks disconnected from the depot that do not serve any required arc. Such solutions with disconnected walks are, however, clearly sub-optimal since the objective function (2a) minimizes the overall travel times. We also observe that constraints (2c) can be obtained as a linear combination of (2e) and (2f). We keep them in the model, however, as in our solution framework not all constraints (2f) are present in the initial formulation, see Section 5.

We next discuss a generalized variant of connectivity constraints (2g) that will allow us to reinforce the dual bound obtained from the LP relaxation of formulation (2). The basic idea of the resulting connectivity constraints (3) is to consider multiple serving variables associated with a particular required arc on the right-hand side, i.e.,

$$X(\delta^-(S)) \geq \sum_{u_p \in S} y^p_{uv} \qquad \forall (u,v) \in A_{\mathrm{R}}, S \subseteq \mathcal{V} \setminus \{0_Q\}. \tag{3}$$

Constraints (3) are valid since equations (2e) imply that the right-hand side is at most one and since arc $(u,v) \in A_{\mathrm{R}}$ can only be served with an initial SOC of p if a walk from the depot 0_Q to node u_p exists.

Lifted connectivity constraints Some of the coefficients in the connectivity constraints (3) can be down-lifted to zero by exploiting a slightly different interpretation of the serving variables. To this end, a variable y^p_{uv} may only be equal to one if the solution contains a walk in $\mathcal{G}$ in which the SOC before the *first traversal* of $(u,v) \in A_{\mathrm{R}}$ is equal to p. Thus, if $y^p_{uv} = 1$ the walk of $\mathcal{G}$ connecting 0_Q with u_p must not contain any arc from $\mathcal{A}_{uv}$. This is formally captured by lifted connectivity constraints (4) in which the coefficients of the arcs from $\delta^-(S)$ that must not be used are down-lifted to zero.

$$X\left(\delta^-(S) \setminus \mathcal{A}_{uv}\right) \geq \sum_{u_p \in S} y^p_{uv} \qquad \forall (u,v) \in A_{\mathrm{R}}, S \subseteq \mathcal{V} \setminus \{0_Q\}. \tag{4}$$

Figure 3c shows a solution to the LP relaxation of formulation (2) of the instance given in Figure 3a that satisfies connectivity constraints (3). Variables y are equal to zero except for $y^1_{23} = y^3_{23} = 0.5$. This solution

violates a lifted inequality (4) for required arc $(2,3)$ and $S = \{2_3, 3_0\}$ since arc $(2_1, 3_0)$ is excluded from the set of cut arcs resulting in a minimum cut of zero.

Special cases of connectivity constraints A particular special cases of connectivity constraints (3) arise if set S contains either all copies of a node in V or no copy at all. That is, for $S = \mathcal{V}(W) = \{u_p \in \mathcal{V} \mid u \in W\}$ with $W \subset V \setminus \{0\}$ such that $W \cap \{u \in V : (u,v) \in A_{\mathrm{R}}\} \neq \emptyset$, connectivity constraint (3) can be re-written as

$$X(\delta^-(\mathcal{V}(W))) \geq 1 \qquad \forall W \subseteq V \setminus \{0\} : W \cap \{u \in V : (u,v) \in A_{\mathrm{R}}\} \neq \emptyset. \tag{5}$$

These cuts correspond to well-known connectivity constraints in the original graph G [3] and have the benefit that they can be separated in G instead of the relatively large energy-indexed graph $\mathcal{G}$.

The ILP formulation presented above requires the implementation of a branch-and-cut algorithm, as it contains several families of constraints of exponential size, whose separation will be discussed in Section 5.2. Since the computational performance of such an exact solution method can be highly affected by the quality of solutions used to initialize the upper bounds of the branching tree, in Section 4 we will also propose efficient ways for constructing high-quality solutions.

3 Solution representation and decoding

In this section we propose two possible ways of encoding feasible solutions. One way is to store the set of traversed arcs for each vehicle, and the other one is to store the set of served arcs per vehicle only. We discuss pros and cons for both options, examine the computational complexity of decoding the optimal travel times in both cases and provide the underlying labeling algorithms.

3.1 Encoding as a sequence of traversed arcs

The definition of eARP solutions includes information on traversed arcs together with chosen travel times. For a given a set of walks, two natural questions arise: 1) Can we determine the battery feasibility of a walk in polynomial time? and 2) If a walk is battery feasible, can we obtain the optimal travel times for each traversed arc in polynomial time? We refer to the latter problem as the *Travel Time Assignment problem (TTAP)*. In the following we denote by $\mathcal{W} = (a^1, a^2, \ldots, a^k)$, $a^j \in A, 1 \leq j \leq k$, a walk starting and ending at the depot for a given eARP instance.

Theorem 1. *Checking whether a walk $\mathcal{W}$ is feasible with respect to the battery capacity Q can be done in polynomial time.*

Proof. For each arc $a^j \in \mathcal{W}$, it is sufficient to consider the travel time leading to the highest SOC after traversing it. More precisely, we use the following recursive formula to calculate the highest SOC b^k after walk $\mathcal{W}$:

$$\begin{aligned} b^0 &= Q, \\ b^j &= \max_{t \in T(a^j)} \beta(a^j, t, b^{j-1}), \quad j \in \{1, \ldots, k\}. \end{aligned}$$

If there exists some $j \in \{1, \ldots, k\}$ with $b^j = -\infty$, the chosen sequence of arcs is not battery feasible, and hence the walk $\mathcal{W}$ is infeasible. This check can be done in polynomial time, namely $\mathcal{O}(k)$, assuming that the max operation can be performed in constant time, e.g., by sorting the travel times during preprocessing. □

On the contrary, finding an optimal travel time for each arc in a feasible walk turns out to be more complicated. As shown in Theorem 2, the TTAP is actually NP-hard.

Theorem 2. *The TTAP is weakly NP-hard.*

Proof. To prove this result, we will provide a polynomial time reduction from the knapsack problem (KP). Consider a KP instance $\mathcal{I}_{\text{KP}}$ associated with a set of items $I = \{1, \ldots, k\}$, with profits p_j and weights w_j, $j \in I$, and knapsack budget Q. The KP asks for a subset of items of maximum total profit whose overall weight does not exceed Q. Let us construct a TTAP instance $\mathcal{I}_{\text{TTAP}}$ that solves $\mathcal{I}_{\text{KP}}$. For this we define an auxiliary graph $G' = (V', A')$ where the node set is given by $V' = \{u^j\}_{j \in I}$. Thus, we associate nodes u^j, $j \in I$, with items of the KP. The walk $\mathcal{W}$ contains arcs $a^j = (u^{j-1}, u^j)$, with $j \in I$ and $u^0 = u^k$. For each connection a^j, $j \in I$, we define two possible travel times $T(a^j) = \{P, P - p_j\}$ where $P = \sum_{j \in I} p_j$, whose associated energy consumption values are equal to zero and to the item's weight, respectively, i.e., for $b \in B$ we have $\beta(a^j, P, b) = b$, while $\beta(a^j, P - p_j, b) = b - w_j$ holds in case $b \geq w_j$, and $\beta(a^j, P - p_j, b) = -\infty$ otherwise. For a given sequence of travel times $\mathcal{T} = (t^1, t^2, \ldots, t^k)$, let $I_{\mathcal{T}} = \{j \in I \mid t^j = P - p_j\} \subseteq I$ denote the subset of items corresponding to the arcs $a^j \in A'$ with travel times are $P - p_j$. Then, the total travel time of sequence $\mathcal{T}$ is $kP - \sum_{j \in I_{\mathcal{T}}} w_j$. Since the energy consumption on each arc a^j with $j \in I_{\mathcal{T}}$ is w_j and zero for all other arcs, $\mathcal{T}$ is battery feasible if and only if $\sum_{j \in I_{\mathcal{T}}} w_j \leq Q$. Hence, finding a sequence of battery feasible travel times that minimizes the total travel time is equivalent to finding an optimal solution to KP. Therefore, the value of an optimal KP solution is equal to K^* if and only if the total travel time of a battery feasible sequence of travel times $\mathcal{T}$ is equal to $kP - K^*$, and the result follows. □

Indeed, Theorem 3 shows that identifying an optimal set of travel times for a given walk can be done in pseudo-polynomial time. This result is particularly useful for two-phase heuristic solution methods that first estimate and fix a walk before computing optimal travel times for the walk's arcs in a second phase by using a dynamic program based on the recursive formula given in the proof of Theorem 3.

Theorem 3. *Let $\mathcal{W}$ be a given walk and $\hat{T} = \max\{|T(a)| : a \in \mathcal{W}\}$ be the maximum number of possible travel times among all arcs of $\mathcal{W}$. Then, an optimal sequence of travel times $\mathcal{T} = (t^1, t^2, \ldots, t^k)$ minimizing the total travel time $\sum_{j=1}^{k} t^j$ such that $(\mathcal{W}, \mathcal{T})$ is battery feasible can be computed in $\mathcal{O}(kQ\hat{T})$.*

Proof. Indeed, it is easy to observe that the minimum travel time $\tau(u^j, b)$ that can be achieved when arriving at node u^j with SOC b can be calculated using the dynamic programming recursion

$$\tau(u^j, b) = \min_{t \in T(a^j), b' \in B:\ \beta(a^j, t, b') = b} \{\tau(u^{j-1}, b') + t\}, \quad \forall j = 1, \ldots, k, \forall b \in B$$

where $\tau(u^0, Q) = 0$ and $\tau(u^0, b) = \infty$ for all $b \neq Q$. The theorem follows since only time-optimal partial solutions may yield an overall solution of minimum total travel time. □

3.2 Encoding as a sequence of served arcs

Alternatively, we propose to represent each feasible solution by storing for each vehicle only a sequence of required arcs served by this vehicle, similar to Vidal [42]. That way, a sequence $\mathcal{R} = (a^1, a^2, \ldots, a^r)$, $a^j \in A_{\text{R}}$, $j \in \{1, 2, \ldots, r\}$, defines which required arcs are served in which order by a particular vehicle. A set of at most m sequences may encode a feasible solution if each required arc is contained in precisely one sequence. Again, we may ask two questions: 1) Given a sequence $\mathcal{R}$, how can we figure out whether it leads to a feasible solution, and 2) If the sequence $\mathcal{R}$ is feasible, how can we compute the missing optimal subwalks between any two consecutively served arcs, as well as associated optimal travel times along the route?

Theorem 4. *For a given sequence $\mathcal{R} = (a^1, a^2, \ldots, a^r)$ of required arcs, checking whether there exists a feasible walk starting and ending at the depot visiting them in the given order can be performed in $\mathcal{O}(r|A||B|)$ time.*

Proof. Starting from the depot, we search for a walk that results in the highest possible SOC at the source node of a^1. To this end, it is sufficient to consider the travel time $\bar{t}(a) \in T(a)$ which consumes the least energy for each arc $a \in A$. The labeling algorithm starts with SOC Q at the depot while the SOC labels at all other nodes are set to $-\infty$. Then, we update the labels of all neighbors v on outgoing arcs $a = (0, v)$ according to $\beta(a, \bar{t}(a), Q)$. The labels of all nodes in the graph are then iteratively updated as long as a label

Algorithm 1: ComputeOptimalWalkForSequence($R, t_{\max}$)

Input: sequence $R = (a^1, a^2, \ldots, a^r)$ with $a^j = (u^j, v^j) \in A_{\mathrm{R}}, j = 1, \ldots, r$, travel time limit $t_{\max}$
Output: optimal travel time t^*, walk $\mathcal{W}^*$, arc travel times $\mathcal{T}^*$
// labels $(t, b, \mathcal{W}, \mathcal{T})$ correspond to non-dominated walks with travel time t, final SOC b, walk $\mathcal{W}$ and travel time sequence $\mathcal{T}$
$\mathcal{L}[v^0] = \{(0, Q, \emptyset, \emptyset)\}$ // initial label for depot $v^0 := 0$
$t_{\max} = t_{\max} - \sum_{a \in A_{\mathrm{R}}} \min_{t \in T(a)} t$
foreach $j = 1, 2, \ldots, r$ **do**
  $\mathcal{L}[u^j] = \text{ComputeParetoOptimalWalks}(\mathcal{L}[v^{j-1}], u^j, t_{\max})$ // compute efficient labels to u^j
  $t_{\max} = t_{\max} + \min_{t \in T(a^j)} t$
  $\mathcal{L}[v^j] = \emptyset$
  foreach $(t, b, \mathcal{W}, \mathcal{T}) \in \mathcal{L}[u^j]$ **do** // extend labels at u^j along $a^j = (u^j, v^j)$
    foreach $t' \in T(a^j)$ **do**
      if $t + t' \leq t_{\max}$ *and* $\beta(a^j, t', b) \in B$ **then** $\mathcal{L}[v^j] = \mathcal{L}[v^j] \oplus (t + t', \beta(a^j, t', b), \mathcal{W} \circ a^j, \mathcal{T} \circ t')$
$\mathcal{L} = \text{ComputeParetoOptimalWalks}(\mathcal{L}[v^r], 0, t_{\max})$ // extend labels to depot
if $\mathcal{L} = \emptyset$ **then return** infeasible
$(t^*, b^*, \mathcal{W}^*, \mathcal{T}^*) = \operatorname{argmin}_{(t,b,\mathcal{W},\mathcal{T}) \in \mathcal{L}} t$ // find label with smallest travel time
return $t^*, \mathcal{W}^*, \mathcal{T}^*$

Algorithm 2: ComputeParetoOptimalWalks($\mathcal{L}, w, t_{\max}$)

Input: set $\mathcal{L}$ of non-dominated labels $(t, b, \mathcal{W}, \mathcal{T})$, target node $w \in V$, travel time limit $t_{\max}$
Output: set $\mathcal{L}$ of non-dominated labels $(t, b, \mathcal{W}, \mathcal{T})$ with target w
foreach *unprocessed label* $(t, b, \mathcal{W}, \mathcal{T}) \in \mathcal{L}$ **do**
  foreach $a \in \delta^+(u)$ *and* $t' \in T(a)$ **do** // node u denotes the final node in walk $\mathcal{W}$
    if $t + t' \leq t_{\max}$ *and* $\beta(a, t', b) \in B$ **then** $\mathcal{L} = \mathcal{L} \oplus (t + t', \beta(a, t', b), \mathcal{W} \circ a, \mathcal{T} \circ t')$
return $\{(t, b, \mathcal{W}, \mathcal{T}) \in \mathcal{L} \mid \mathcal{W} = (a^1, a^2, \ldots, (u^k, w))\}$

with higher SOC than the current one is obtained. That way, $\mathcal{O}(|A||B|)$ time is needed to find out whether a feasible walk between depot 0 and the source node of arc a^1 exists. If this is the case, the labeling returns the highest possible SOC b^1 for the source node of a^1. The SOC label at the target node of a^1 is then set to $\beta(a^1, \bar{t}(a^1), b^1)$ and the process is repeated until all required arcs in $\mathcal{R}$ are processed and the final walk to the depot is found. If the resulting SOC at any of the arcs in $\mathcal{R}$ or at the depot is $-\infty$, the sequence does not admit a feasible walk, otherwise the sequence $\mathcal{R}$ is feasible. □

Algorithm 1 calculates an optimal walk $\mathcal{W}^*$ with the associated sequence of travel times $\mathcal{T}^*$ for a given sequence $\mathcal{R}$ such that the total travel time is minimal. Note that the walk connecting two consecutive required arcs depends on the SOC after traversing the previous required arc. Thus, it cannot be pre-computed as resource constrained shortest path with energy as resource and travel times as costs [24, 28]. To obtain an optimal walk for sequence $\mathcal{R}$ we, therefore, compute at each iteration $j = 1, \ldots, r$, all *non-dominated* labels $(t, b, \mathcal{W}, \mathcal{T})$ from the depot (if $j = 1$) or from v^{j-1}, the target node of the previous required arc $a^{j-1} = (u^{j-1}, v^{j-1})$ (if $j > 1$), to u^j, the source node of required arc $a^j = (u^j, v^j)$, using Algorithm 2 that will be described below.

Each label $(t, b, \mathcal{W}, \mathcal{T})$ is represented by its associated walk $\mathcal{W} = (a^1, a^2, \ldots, a^k)$, $a^l = (w^{l-1}, w^l)$ with $w^0 = 0$, travel time sequence $\mathcal{T} = (t^1, t^2, \ldots, t^k)$, $t^l \in T(a^l)$ for $l \in \{1, 2, \ldots, k\}$, total travel time $t = \sum_{l=1}^{k} t^l$, and final SOC $b \in B$ at node w^k, see Equation (1). A label $(t, b, \mathcal{W}, \mathcal{T})$ is dominated by another label $(t', b', \mathcal{W}', \mathcal{T}')$ whose walk ends at the same node w^k and that serves the same partial sequence of required arcs if and only if $t \geq t'$ and $b \leq b'$. Algorithms 1 and 2 use operators $\circ$ and $\oplus$. We apply $\circ$ to append an

arc to a walk and also to append a travel time to a sequence of travel times. Operator $\oplus$ is used for possibly inserting a new label: If the new label is non-dominated, it is inserted and all labels dominated by it are removed. This can be done in time linear in the number of labels for one particular node.

The first iteration of Algorithm 1 starts at the depot and computes all non-dominated labels $(t, b, \mathcal{W}, \mathcal{T})$ to u^1. In iteration j, the previously found labels are extended to non-dominated labels that correspond to walks to the source of the j-th required arc in the sequence. All non-dominated labels are then extended by traversing the required arc (u^j, v^j) for all possible travel times. After processing the whole sequence, the resulting labels are extended to the depot, so that finally each label corresponds to a Pareto-optimal walk that starts and ends at the depot. Among those, the one with minimal total travel time is returned. Labels from set $\mathcal{L}$ are processed in a FIFO fashion.

Algorithm 2 is used as a subroutine in Algorithm 1. Given a set of labels $\mathcal{L}$, it computes all non-dominated labels that correspond to walks ending at a given target node w and that can be obtained by extending one of the labels from $\mathcal{L}$. In each iteration, an unprocessed label $(t, b, \mathcal{W}, \mathcal{T})$ is chosen from set $\mathcal{L}$ and extended by traversing all arcs emanating from its end node with all associated travel times that are battery feasible. This process is repeated as long as $\mathcal{L}$ still contains unprocessed labels. Finally, all labels corresponding to walks ending at the given target node w are returned.

Both algorithms use a given maximum total travel time $t_{\max}$ to restrict the label extension. Observe that this overall bound can be tightened to $t_{\max} - \sum_{a \in A_R \setminus \{a^1, a^2, \ldots, a^{j-1}\}} \min T(a)$ in iteration j in which the visited required arcs correspond to the sequence $(a^1, a^2, \ldots, a^{j-1})$, see lines 2 and 5 of Algorithm 1.

Theorem 5. *For a given sequence $\mathcal{R}$ of required arcs, Algorithm 1 obtains a feasible walk $\mathcal{W}$ with minimal total travel time (if it exists).*

Proof. Observe that the problem can be seen as a non-elementary shortest path problem with resource constraints (NESPPRC) on a larger graph which is obtained by replicating graph G $r + 1$ times. In this transformation, the source is the depot in the first copy of G, the j-th copy of G is connected to the $(j+1)$-th copy through the required arc $a^j \in \mathcal{R}$, and the target is the depot in the $(r+1)$-th copy of G. The cost of the walk is the travel time, and the resource is the SOC at every node. Instead of solving the NESPPRC on such a large graph, Algorithm 1 offers a memory-efficient implementation of the NESPPRC algorithm on G. The usual labeling algorithm for the NESPPRC produces labels for each node $u \in V$ in each of the $r + 1$ copies of G. A label of u in the j-th copy of G corresponds to a label of u in the j-th iteration of our algorithm. This follows from the fact that the reset of the previous labels for all $u \in V \setminus \{v^{j-1}\}$ happens in Step 4 by keeping only the labels for v^{j-1}, the target of arc a^{j-1}, which are carried over to the next iteration. □

4 Heuristics

We address in the following several efficient ways for constructing feasible routes, which can be used either in the initial phase of the branch-and-cut algorithm (cf. Section 5), or as stand-alone approaches for dealing with large-scale instances. All construction and improvement heuristics proposed in Section 4.1 solely operate on the solution representation presented in Section 3.2. Several acceleration techniques are introduced in Section 4.2.

4.1 Construction and improvement heuristics

We propose three different construction heuristics for the eARP all of which are based on representing a solution as a sequence of required arcs and use Algorithm 1 for solution decoding and evaluation.

Heuristic *SM* This heuristic is inspired by the Clarke-Wright savings heuristic for the capacitated VRP [12] and the Merge heuristic for the capacitated ARP [20]. Starting with a set of sequences where each contains a single required arc, the heuristic iteratively concatenates two sequences while preserving feasibility with respect to the battery capacity. Note that we do not consider merge possibilities resulting from reversing one (or both) of the sequences of required arcs. In preliminary experiments it turned out that

evaluating the latter options requires a lot of time with no significant avail. A best improvement strategy that selects and merges two sequences maximizing the resulting decrease of the overall travel time is used in each iteration. Ties are broken by choosing the two sequences which result in the longest sequence to prioritize the building of a few long routes. The heuristic terminates when the number of sequences is not larger than the number of vehicles and any merge of two remaining sequences would increase the travel time. Thus, merges that increase the objective function value may be performed in case no time-decreasing merge exists and the intermediate solution still contains more than m sequences.

Heuristic *RC* This heuristic follows a route-first-cluster-second approach [5]. It first computes a giant route that minimizes the total travel time and includes all required arcs by ignoring the battery capacity. To this end, we observe that an optimal solution to a relaxation of the eARP that neglects the battery constraints can be computed by solving an instance of the DRPP on G in which only the fastest travel time from $T(a)$ is considered for each arc $a \in A$. An optimal solution to the DRPP is obtained by using a branch-and-cut algorithm based on a natural-space formulation using connectivity cuts that ensure a directed path from the depot to every required arc and vice versa [11]. This approach turned out to be feasible since the DRPP considering only the lowest travel times can be solved extremely fast even for the most difficult eARP instances considered in our study. Computing an optimal DRPP solution also has the benefit of providing a valid lower bound on the optimal solution value of the associated eARP instance since the DRPP is a relaxation of eARP. Moreover, if the obtained DRPP solution is battery feasible it is an optimal solution to the eARP instance as well. In general, however, the giant route needs to be split into multiple routes. For this, the visiting sequence of all required arcs is derived from the giant route before splitting the latter in an optimal way using a dynamic program similar to the one proposed for the capacitated ARP in Lacomme et al. [25] that is detailed in Algorithm 3. For each $k, l \in \{1, 2, \ldots, |A_{\mathrm{R}}|\}$ the optimal walk for the subsequence $(a^k, a^{k+1}, \ldots, a^l)$ is computed by Algorithm 1. The combination of the best set of sequences for $(a^1, a^2, \ldots, a^{k-1})$ together with the optimal walk for $(a^k, a^{k+1}, \ldots, a^l)$ is stored as a new best solution covering all required arcs up to a^l in case it is better (i.e., faster) than the so-far best set of walks covering these arcs. Variables p are used to efficiently keep track of the indices of required arcs at which the giant route is split. Merge heuristic *SM* is finally applied to the result of Algorithm 3 in case the number of derived routes exceeds the given number of vehicles m. Each split sequence of required arcs is optimally decoded by Algorithm 1, see Theorem 5. Thus, Corollary 1 follows from arguments analogous to those used in [25, 41] for the capacitated ARP.

Corollary 1. *The set of walks obtained by the split algorithm is optimal for the given sequence of required arcs if the number of obtained routes does not exceed m.*

Heuristic *CM* This heuristic combines ideas from the previous two heuristics. First, the weakly connected components of the graph induced by all required arcs are determined. Then, heuristic *RC* is applied to each of these components. The resulting set of sequences is then used as initial solution to heuristic *SM* (rather than routes containing only a single required arc).

Improvement Solutions computed by any of these three heuristics are subsequently improved by a local search algorithm that moves a single required arc to another position (either within its sequence or to an arbitrary position of any other sequence). Again, a best improvement strategy is used and the local search algorithm stops if there is no move that improves the objective function value.

4.2 Acceleration techniques

Algorithm 2 may induce long running times in case many Pareto-optimal walks exist and must be processed. Three techniques that aim to address this issue are considered in our heuristic algorithms: (i) an upper bound on a walk's travel time, (ii) a limit on the number of kept labels per node, and (iii) a sequence archive to re-use already evaluated labels.

Algorithm 3: SplitGiantRoute($R, t_{\max}$)

Input: sequence $R = (a^1, a^2, \ldots, a^{|A_R|})$ with $a^j \in A_R, j = 1, 2, \ldots, |A_R|$, travel time limit $t_{\max}$
Output: set $\mathcal{R}$ of sequences

$t_0 = 0$
$t_k = \infty,\ \forall k = 1, 2, \ldots, |A_R|$ // travel time of best split up to a_k
$p_k = 0,\ \forall k = 0, 1, \ldots, |A_R|$ // reference to previous best split point
for $k = 1, \ldots, |A_R|$ **do** // determine optimal split points
  for $l = k, \ldots, |A_R|$ **do**
    $t^*, \mathcal{W}^*, \mathcal{T}^* =$ ComputeOptimalWalkForSequence($(a^k, a^{k+1}, \ldots, a^l), t_{\max}$)
    // optimal walks for a^1 to a^{k-1} plus $\mathcal{W}^*$ better than best known up to a^l?
    if $t_{k-1} + t^* < t_l$ **then** $t_l = t_{k-1} + t^*,\ p_l = k$
$\mathcal{R} = \emptyset,\ k = |A_R|$
while $k \neq 0$ **do** // derive optimal set of sequences
  $\mathcal{R} = \mathcal{R} \cup \{a^{p_k}, a^{p_k+1}, \ldots, a^k)\}$
  $k = p_k - 1$
return $\mathcal{R}$

As detailed in Section 3.2, an upper bound $t_{\max}$ on the travel time of a walk is used to limit the extension of labels. We set $t_{\max}$ equal to the objective value of the best solution found so far (if available).

To further limit the number of labels, we only keep the best K labels for each node with respect to the walk's travel time. With this modification we significantly reduce the running times, but also lose the optimality property of the labeling algorithm. Our experimental results show that the decrease in solution quality is only moderate and often negligible even when using comparably small values of $K \leq 20$.

If two different sequences of required arcs have a common prefix the set of Pareto-optimal walks is identical up to the point where the sequences diverge. Thus, we accelerate our heuristic algorithms by re-using results from previous sequence decodings. Sets of Pareto-optimal walks for past sequences are efficiently stored using a solution archive based on a trie structure in which each node corresponds to a single required arc and its children are associated with succeeding required arcs in already considered sequences, see Raidl and Hu [37], Ruthmair and Raidl [38]. This data structure allows to efficiently insert sets for newly computed sequences and find the longest already computed prefix for a given sequence. Thus, Algorithm 1 can skip the required arcs contained in the found prefix and hot-starts from the set $\mathcal{L}$ of Pareto-optimal walks stored for this prefix.

5 Algorithmic framework

All formulations and algorithms described in the previous sections have been implemented in C++ using the branch-and-cut framework IBM ILOG CPLEX 12.9 with default settings. In this section we discuss aspects that influence the results of our computational study. These include the generation of the energy-indexed graph, rules for reducing its size, and algorithmic details of the separation methods used for dynamically adding violated inequalities as cutting planes.

5.1 Energy-indexed graph generation

As mentioned in Section 2, creating the energy-indexed graph $\mathcal{G}$ by simply replicating all nodes and arcs for all possible SOCs in B would induce a huge graph containing many states (i.e., energy-indexed nodes or arcs) which cannot be obtained in an eARP solution. Instead, we incrementally create $\mathcal{G}$ by only considering node-SOC pairs that can appear in an energy-feasible walk starting from the depot with SOC level Q. We maintain a queue of energy-indexed nodes initially containing only 0_Q. For each node in the queue we add

one energy-indexed arc for each outgoing arc and travel time. If some end node of a new arc does not exist yet we add it to $\mathcal{G}$ and the queue. After this generation procedure the energy-indexed graph is reduced by removing nodes and arcs that cannot appear in a feasible or optimal solution. The following three rules are applied repeatedly until no further reductions can be made:

1. Nodes without outgoing arcs cannot be part of a closed walk and are thus removed together with all incoming arcs.

2. Cycles containing only optional arcs whose traversal decreases the SOC cannot be part of an optimal solution. Some two-cycles of this kind can be eliminated with the following rule: Let $(u, v), (v, u) \in A \setminus A_{\mathrm{R}}$ be two optional arcs of G. Then, energy-indexed arc $(u_p, v_q) \in \mathcal{A}$ can be removed if $(v_q, u_{p'}) \in \mathcal{A}$ is the only arc going out of v_q and $p' \leq p$, or if $(v_{q'}, u_p) \in \mathcal{A}$ is the only arc going into u_p and $q' \geq q$.

3. Nodes without incoming arcs might appear due to the previous rule. Such nodes are removed together with all outgoing arcs.

5.2 Separation of valid inequalities

Preliminary experiments indicated computational advantages of introducing integer variables $x_{uv} \in \mathbb{Z}^+$ for all arcs $(u, v) \in A$, flow conservation constraints defined on these variables similar to (2d) for each $u \in V$, and linking constraints $x_{uv} = X(\mathcal{A}_{uv})$. While these variables (and constraints) are clearly redundant, branching on them may improve the balance of the resulting branch-and-cut trees. As detailed below, they are also useful when separating connectivity constraints (5).

The following order is used to separate the different classes of connectivity constraints: (i) constraints (5) whose cutsets contain all or no energy-indexed copies of nodes $u \in V$, which can be rewritten with x variables and thus separated in original graph G; (ii) general energy-indexed graph connectivity constraints (3) or their stronger variants (4). Connectivity constraints (2g) are not considered in the solution framework since they are dominated by (3) and (4), and the separation in general involves more computational time.

Note that not all types of connectivity constraints are considered in each algorithmic variant tested in our computational study, and that a particular type is only considered in a current call of the separation algorithm if no violated inequalities of all previous types (according to the list above) have been identified. Layered graph connectivity constraints (3) and (4) are only added if the *cut violation* ratio between left-hand side and right-hand side value is smaller than 0.9. This strategy turned out to be a good compromise in preliminary experiments to avoid stalling and separation of shallow cuts at fractional points and still improve the LP bounds. Based on these results, we also decided to separate strengthening cuts (of all classes) for fractional candidate solutions only at the root node of the branch-and-cut tree with a limit of at most 100 inequalities in each iteration. In the following paragraphs we describe how we exactly identify violated inequalities (3)-(5) for fractional solutions. Let $\bar{x}$, $\bar{X}$, and $\bar{y}$, respectively, denote the variable values of the current LP relaxation before entering the separation procedures.

Separation of constraints (3) and (4). Using equations (2e) we can rewrite constraints (3) as

$$X(\delta^-(S)) \geq 1 - \sum_{u_p \notin S} y^p_{uv} \qquad \forall (u, v) \in A_{\mathrm{R}}, S \subseteq \mathcal{V} \setminus \{0_Q\},$$

which is equivalent to

$$X(\delta^-(S)) + \sum_{u_p \notin S} y^p_{uv} \geq 1 \qquad \forall (u, v) \in A_{\mathrm{R}}, S \subseteq \mathcal{V} \setminus \{0_Q\}. \tag{6}$$

Hence, we construct the support graph for finding a minimum cut for a given required arc $(u, v) \in A_{\mathrm{R}}$ as follows: We introduce an artificial target node t and connect all sources of the arcs $\mathcal{A}_{uv}$ to t, i.e., we add arcs $\{(u_p, t) : \exists (u_p, v_q) \in \mathcal{A}_{uv}\}$. The capacities of arcs (u_p, t) are set to $\bar{y}^p_{uv}$ and for all other arcs in $\mathcal{A}$ to

their $\bar{X}$ value. Any $0_Q - t$ cut in this support graph whose value is less than one corresponds to a violated inequality (6). We define set $S \subset \mathcal{V}$ to be the set of nodes from $\mathcal{G}$ in the cut partition containing t and hence the set of cut-arcs for set S leads exactly to the left-hand side of (6). To separate the lifted variant (4) for one particular required arc $(u, v) \in A_{\mathrm{R}}$, the only difference to the procedure above is that the capacity of all its energy-indexed copies $\mathcal{A}_{uv}$ are set to zero in the support graph.

Since the number of linking constraints (2f) connecting y and X variables can be rather high, we do not include them a priori in the formulation but add them dynamically only for those y variables that arise on the right-hand side of some violated cut (3) or (4) and only if the corresponding linking constraints are violated, too. Linking constraints (2f) violated by cuts added in previous iterations are also added if the current LP solution does not violate inequalities (3) and (4).

Separation of constraints (5)**.** Violated connectivity constraints (5) can be identified in the original graph G since all copies $u_p \in \mathcal{V}$ of some node $u \in V$ must be either in set S or not. For each node $u \in V$ which is a source of some required arc, i.e., $\delta^+(u) \cap A_{\mathrm{R}} \neq \emptyset$, we use values $\bar{x}$ as arc capacities and compute a maximum flow in G from node 0 to node u. If the capacity of a minimum cut is below one we found a violated inequality (5).

In some cases multiple minimum cuts can be found. To obtain one with a minimum number of cut arcs (leading to a sparser inequality) we add a small value, i.e., 10^{-5}, to all arc capacities, cf. [17]. If there are still multiple minimum cuts we choose a cut which has a smallest set S.

When we are faced with an integer solution in the separation of inequalities (3)-(5), we use a more efficient strategy based on breadth-first search in the support graph. For inequalities (3) and (4), we only add a single violated inequality (plus necessary linking constraints (2f) if applicable) to cut-off infeasible solutions.

6 Experimental results

In this section we first describe and motivate our test instances, followed by a comparison and performance evaluation of our branch-and-cut and heuristic algorithms, respectively. Finally, we perform extended experiments with varying battery sizes and street network properties to derive managerial insights and provide recommendations. Each experiment has been performed on a single core of an Intel Xeon E5-2670v2 machine with 2.5 GHz. A memory limit of 8 GB has been set for each test run.

6.1 Instances

As a guideline for constructing reasonable benchmark instances we consider long-range arc routing applications on primary streets, e.g., highways, in which the vehicle's load capacity can be ignored. This is because for inner city areas the battery capacity of currently available electric trucks, see, e.g. [15], seems to be more than sufficient to allow a replacement of trucks with combustion engine without reaching the range limit within a single day provided that there are overnight re-charging facilities at central depots.

The street networks are obtained by using the arc routing instance generator OARBench [30]. We chose three cities in Europe—Amsterdam (A), Paris (P), Vienna (V)—and selected all primary streets within a predefined rectangle around the center (Amsterdam: 16.4×17km, Paris: 13.5×12km, Vienna: 18.5×18.5km). Arc lengths are set to Euclidean distances. The resulting large-scale networks with thousands of nodes and arcs are heavily preprocessed to obtain manageable test instances still capturing the city structure: We repeatedly remove all nodes with no outgoing or incoming arcs and all self-loops. In case of parallel arcs (which may also result from other preprocessings) we keep the longest one. Nodes which have exactly two neighbors are removed and the incident arcs are merged. Similarly, a node with a single incoming (outgoing) arc is removed and the arc is merged to all outgoing (incoming) arcs. Finally, we merge nodes which are within a Euclidean distance of 500, 300, or 100 meters, resulting in three graphs with different sizes for each city. The single depot is set near a train station or public service garage. For each of the nine street networks

we define two different sets of required arcs, randomly selecting about 10% and 50% of all arcs, respectively. Instance names start with the first letter of the city followed by $|V|$-$|A|$-$|A_{\mathrm{R}}|$, e.g., A-31-105-8.

We compute the vehicle's energy consumption as described in [2], i.e., by reformulation we obtain for travel time t and speed v,

$$e(t,v) = t\left(\gamma_1 v + \gamma_2 v^3 + \gamma_3\right), \tag{7}$$

where $\gamma_1, \gamma_2, \gamma_3$ are vehicle-specific constants: Based on the specifications of an Eforce EF18 electric truck [15] which is based on an IVECO Stralis, a total constant weight of 15 tons, 3 kW power for auxiliary components (heating, cooling, etc.), and zero street gradients, we obtain

$$\begin{aligned}
\gamma_1 &= \frac{g \cdot C_r \cdot m}{\eta} = \frac{9.81 \cdot 0.008 \cdot 15000}{0.97} \approx 1213.61,\\
\gamma_2 &= \frac{C_w \cdot A \cdot \rho}{2 \cdot \eta} = \frac{0.8 \cdot 9.69 \cdot 1.165}{2 \cdot 0.97} \approx 4.66,\\
\gamma_3 &= P_0 = 3000,
\end{aligned}$$

where g is gravity, C_r rolling resistance, m weight, η engine efficiency, C_w drag coefficient, A frontal surface area, ρ air density (200m above sea level, temperature 20°C), and P_0 auxiliary power. Note that for a fixed speed the energy consumption is linear in time, i.e., $e(t,v) = \hat{e}(v)t$ with $\hat{e}(v) := \gamma_1 v + \gamma_2 v^3 + \gamma_3$.

From equation (7) by replacing t with d/v we can derive the speed to drive some distance d that leads to the minimal energy consumption, i.e., $v^* = \sqrt[3]{\gamma_3/(2\gamma_2)} \approx 24.7$ km/h. With this speed the truck needs about 1870 kWs ≈ 0.52 kWh per kilometer. To account for inaccuracies in the parameters and other uncertainties, SOC values are underestimated by rounding down to the nearest 1000 kWs. We consider four different battery sizes, i.e., $Q \in \{25, 37.5, 50, 75\}$ (in kWh) while the Eforce EF18 can use batteries up to 630 kWh. Since the battery is one of the major factors in the truck acquisition costs we want to keep it as small as possible and it turned out that already small sizes are sufficient in our cases, see Section 6.4.

Assuming a maximal charging rate r, we use a piece-wise linear charging function $c(t,b)$ depending on travel time t and initial SOC level b accounting for the battery's reduced capability of absorbing energy when the SOC is high: From 0% to 85% SOC the battery can be charged with the maximal rate of $r_1 := r$, from 85% to 95% the rate halves to $r_2 := \frac{r}{2}$, and from 95% to 100% the rate reduces to a sixth, i.e., $r_3 := \frac{r}{6}$, cf. [32]. For an empty battery, i.e., $b = 0$, and charging time t, the SOC after charging—assuming that no energy is consumed while charging—is defined by function

$$\hat{c}(t) = \min\{r_1 t,\ 0.85Q + r_2(t - t_1),\ 0.95Q + r_3(t - t_2),\ Q\}, \tag{8}$$

where $t_1 = \frac{0.85Q}{r_1}$ and $t_2 = t_1 + \frac{0.1Q}{r_2}$ are the first two breakpoints of the piece-wise linear, concave, and non-decreasing charging function. For positive SOC levels $b > 0$ function $\hat{c}(t)$ is shifted by $\hat{c}^{-1}(b)$ to the left. Thus, we apply the transformation from [43] to a one-dimensional function, i.e., $c(t,b) = \hat{c}(t + \hat{c}^{-1}(b))$. Note that the inverse $\hat{c}^{-1}(b)$ is uniquely defined for $b \in [0, Q[$ while for $b = Q$ we use the minimum time to obtain a full battery which is the third breakpoint $t_3 = t_2 + \frac{0.05Q}{r_3}$.

We did not consider that driving along a dynamic charging arc also simultaneously consumes energy. Since the battery itself can either be charged or discharged at a time, a battery management unit determines the net energy rate $r - \hat{e}(v)$ in such a situation and decides whether to re-charge (if $r - \hat{e}(v) > 0$) or discharge (if $r - \hat{e}(v) < 0$) the battery. Thus, if $r - \hat{e}(v) > 0$ we adapt the charging rates used in function $\hat{c}(t)$ to speed-dependent values $r_1(v) := r - \hat{e}(v),\ r_2(v) := \min\{r - \hat{e}(v), \frac{r}{2}\},\ r_3(v) := \min\{r - \hat{e}(v), \frac{r}{6}\}$. Note that $r_1(v) > 0$ implies $r_2(v) > 0$ and $r_3(v) > 0$.

Dynamic charging facilities are installed on at most $p_c|A|$ arcs, restricted only to the set of required arcs, with $p_c = 0.1$ if not stated otherwise. The first $\min\{p_c|A|, |A_{\mathrm{R}}|\}$ required arcs (in the order of appearance in the instance) are equipped with wireless chargers with a rate of $r = 22$ kW throughout the full distance d_a of arc $a \in A$. In our default setting, we consider two travel times on non-charging arcs based on speeds of 40 and 60 km/h which we assume to be constant on the whole arc. For charging arcs we add a third option to allow longer re-charging, i.e., 20 km/h. Note that recent wireless charging technology allows to re-charge with 22 kW at any of these speeds [8, 10].

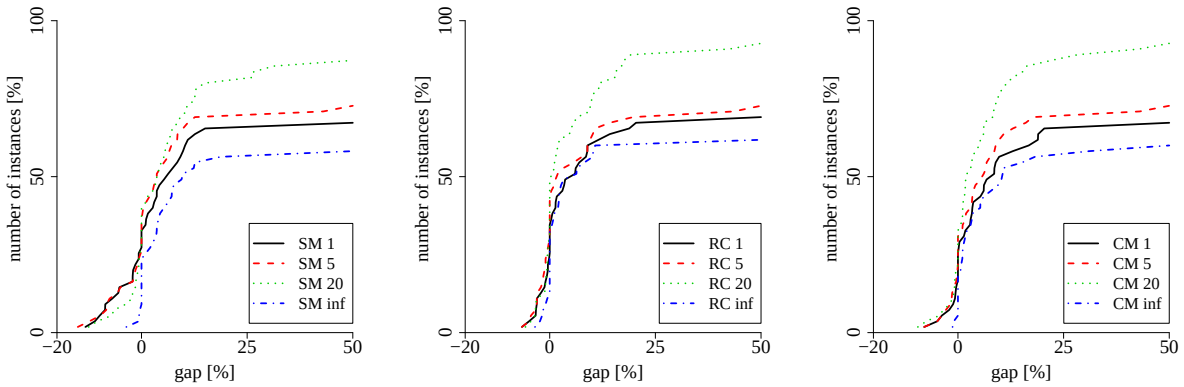


Figure 4: Gaps of heuristic solutions compared to best branch-and-cut solutions.

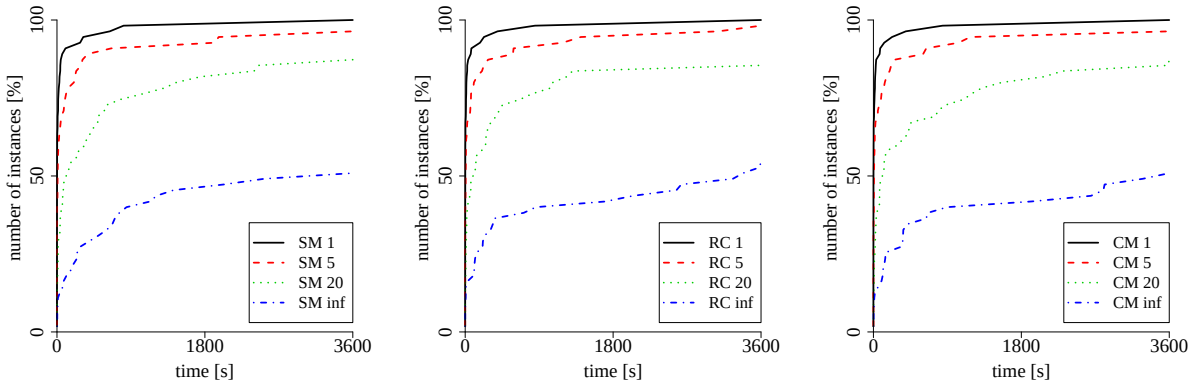


Figure 5: Solution times of heuristics.

Finally, the SOC function $\beta(a,t,b)$ on arc $a \in A$, for travel time t, and initial SOC b is defined as

$$\beta(a,t,b) = \begin{cases} c(t,b) & \text{if } a \text{ is a charging arc and } r_1(\frac{d_a}{t}) > 0, \\ b + r_1(\frac{d_a}{t})t & \text{if } r_1(\frac{d_a}{t}) \leq 0 \text{ and } b + r_1(\frac{d_a}{t})t \geq 0, \\ -\infty & \text{else,} \end{cases} \tag{9}$$

where the first case describes the battery charging situation with a positive net rate, the second case discharges the battery based on a non-positive net rate (note that $r = 0$ on non-charging arcs), and the third case accounts for infeasibility because of a too low initial SOC b.

6.2 Performance of heuristics and the branch-and-cut algorithm

In this section we report the performance of the heuristics given in Section 4 and the exact solution framework described in Section 5 on our set of benchmark instances. We start by evaluating the heuristics and analyze their solution quality in function of the number of labels K kept per node in the labeling algorithm. Then we focus on the savings of the computing time thanks to the usage of the archive. The three construction heuristics SM, RC, and CM, followed by the local search phase are applied to the city instances with a time limit of one hour.

Figure 4 depicts cumulative gaps of the three heuristics for values of $K \in \{1, 5, 20, \infty\}$. Additionally, Figure 5, depicts their cumulative runtimes in seconds. The quality of a heuristic solution is measured as the relative deviation of the travel time with respect to the best solution found by one of the branch-and-cut (BC)

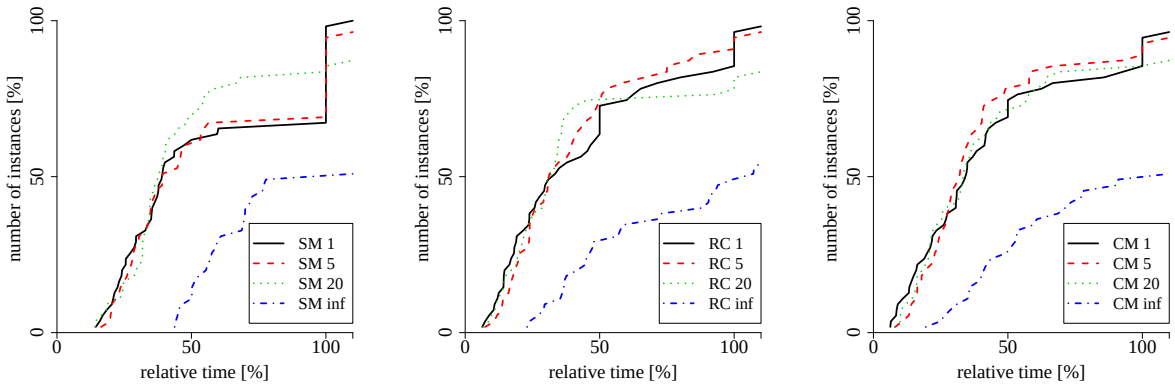


Figure 6: Solution times of heuristics with solution archive relative to variants without archive.

approaches. A point with coordinates (x, y) in these charts indicates that for $y\%$ of instances the obtained deviation from the best BC solution is $\leq x\%$. Negative deviations show that the BC approach is not able to sufficiently improve the initial heuristic solution (for the BC methods we only use construction heuristic RC with label limit $K = 20$ without any local search). A closer look into the results, see Appendix A.1, reveals that negative gaps are obtained for the cases where the energy-indexed graph formulation is too large to be handled efficiently.

We recall that higher values of K in general allow to explore more solutions (especially for low battery sizes, $Q = 25$), since more labels with higher SOC (and longer travel times) are kept for each node. On the contrary, with K being too low ($K \leq 5$) for some instances it was sometimes not possible to find a feasible solution. However, Figure 4 clearly indicates that higher values of K do not necessarily imply a better solution quality. This can be easily explained by the enormous runtime increase for growing values of K due to the additional effort in sequence decoding in Algorithm 1. This sometimes leads to deviations of more than 100% when the time limit is reached before finding reasonably good solutions. The best trade-off is achieved for $K = 20$, which we choose as a default setting for the rest of the experiments.

More details from comparing the heuristics can be found in Appendix A.1. From these runs we observe that the number of weakly connected components induced by required arcs (denoted by C in the second column of Table 3) seems to be a distinguishable factor for choosing the construction heuristic: The route-first-cluster-second heuristic RC often obtains better solutions if the number of components is rather high, i.e., when the relative number of required arcs is low. The initial giant route obtained by solving the DRPP seems to efficiently capture the component structure. Surprisingly, CM mostly performs worse than RC for these cases even though it explicitly exploits the component structure of the graph. Probably, the merge phase after separately considering each component is counterproductive here. On the other hand, the merge-based heuristic SM seems to be more appropriate for more difficult instances with a higher number of required arcs (with less components).

RC is on average faster than the other candidates, mainly because both solving the DRPP and splitting the giant route can be done quite efficiently while evaluating the merge options in SM and also CM takes a considerable amount of time because of the decoding procedure.

Figure 6 depicts cumulative runtimes of the three heuristics when the solution archive is used. Runtimes are given as percentage of the runtime of the same setting without archive. From these charts we can observe significant speedups achieved by using the archive, sometimes even by an order of magnitude. In some rare cases the overhead of managing the archive leads to a slower running time ($> 100\%$). The speedup is on average higher for larger battery sizes for which there is an increased effort for sequence decoding since more feasible labels can be generated. The results also show that RC and CM benefit more from the archive. The reason is that when splitting the giant route in Algorithm 3 the sequences are iteratively extended at the end allowing to re-use the labels stored in the archive from previous iterations.

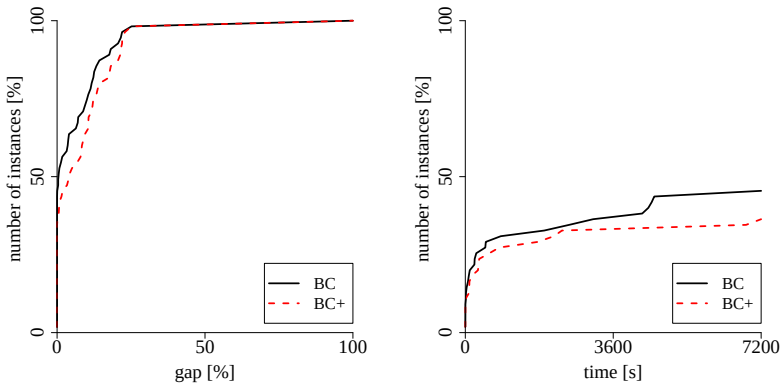


Figure 7: Final optimality gaps and computing times for branch-and-cut approaches.

To conclude, the solution archive clearly enhances the heuristics mainly due to the high efforts in decoding the sequences of required arcs. Moreover, heuristics SM and RC with label limit $K = 20$ together with the solution archive seem to provide a good compromise between solution quality and runtime.

We now turn our attention to the performance of the BC methods and compare the following two settings:

- Variant BC uses the first visit variables y, adds (2e) and (2i) statically to the formulation and separates linking constraints (2f) and lifted inequalities (4) dynamically for integer solutions only.
- Variant BC+ is an extension of BC by separating fractional points as well and adding inequalities (4) to strengthen the dual bound of the LP relaxation (only) at the root node.

Both BC variants use the base formulation (2a)-(2d) together with (2h), and separate connectivity cuts (5) on the original graph G both for integer and fractional solutions. An initial feasible solution is generated by heuristic RC with label limit $K = 20$, enabled solution archive, and disabled local search (to avoid spending too much time in this phase) and handed over to CPLEX as a first incumbent solution (if available).

Table 1 reports for each instance and for three possible battery capacities $Q \in \{25, 50, 75\}$ the following values: the best lower bound (LB) and the best upper bound (UB) on the optimal total travel time in seconds found by the two variants of the BC algorithm. We also report the total runtime in seconds, the number of separated cuts, cf. Section 5, and the total number of branching nodes. For each of the three heuristics, we report the relative gap to the best upper bound UB, obtained with the default setting ($K = 20$, enabled archive and local search). The runtime of each single BC test run is limited by two hours and we mark it with "TL" when the time limit is reached without proving optimality. Entries in boldface indicate the best results for each row and block of columns.

The results indicate that, as could be expected, it is much harder to solve instances with a high number of required arcs, especially for larger batteries. Smaller battery sizes B usually lead to smaller ILP formulations but also to worse LP relaxation bounds, indicated by the higher number of branch-and-bound nodes. When the battery limits increase the energy-indexed graphs and the corresponding formulations quickly get larger as well as the time for solving the associated LP relaxations. Although the LP bounds are quite tight in these cases, the BCs often exceed the time limit ending up with a few processed branch-and-bound nodes only. In general, the base energy-indexed graph formulations with connectivity cuts (5) but without inequalities (4) are already quite strong, as also observed in [21] for layered graph formulations for other problems. The residual optimality gap after two hours is usually below 10% except for the largest instances with many required arcs. In these cases the dual bounds are already quite good but the primal bounds are mostly far from the optimum which indicates the need for a more extensive initial heuristic phase. This is also confirmed by the negative values obtained in the heuristics columns, which are particularly present for $Q \in \{50, 75\}$. We also investigated adaptations of our construction heuristics to be applied in branch-and-bound nodes

Table 1: Comparison of branch-and-cut approaches and heuristics (best values per line are marked **bold**)

		LB		UB		runtime [s]		#cuts		#BB nodes		UB [%] ($K = 20$)		
city-$\|V\|$-$\|A\|$-$\|A_R\|$	Q	Y	Y+	Y	Y+	Y	Y+	Y	Y+	Y	Y+	SM	RC	CM
A-31-105-8	25	**6485**	**6485**	**6485**	**6485**	16	**9**	**409**	699	1419	**284**	**1.4**	**1.4**	1.8
	50	**5032**	**5032**	**5032**	**5032**	**1**	**1**	**0**	**0**	**0**	**0**	**0.0**	**0.0**	**0.0**
	75	**5032**	**5032**	**5032**	**5032**	2	**1**	**0**	**0**	**0**	**0**	**0.0**	**0.0**	**0.0**
A-31-105-47	25	**30247**	**30247**	**30247**	**30247**	**15**	154	**1246**	2317	23	**0**	**4.6**	9.9	8.2
	50	**19253**	19044	**19324**	19806	TL	TL	**5137**	11067	7876	**335**	7.1	2.2	**1.1**
	75	**18289**	**18289**	**18298**	19868	TL	TL	**8285**	12356	525	**0**	8.6	**2.1**	**2.1**
A-58-168-15	25	**11329**	**11329**	**11329**	**11329**	**85**	96	**803**	1596	1587	**0**	**0.0**	**0.0**	1.2
	50	**7770**	7564	**7770**	7966	**4454**	TL	**1392**	6629	2124	**33**	**0.0**	0.5	**0.0**
	75	**6733**	**6733**	**6733**	**6733**	**2**	**2**	**0**	**0**	**0**	**0**	**0.0**	**0.0**	**0.0**
A-58-168-81	25	**41982**	**41982**	**41982**	**41982**	**267**	586	**3873**	6408	801	**73**	**3.2**	9.6	9.6
	50	**24113**	24081	**25059**	28153	TL	TL	**5183**	13226	591	**0**	6.0	**5.1**	**5.1**
	75	**22480**	22478	**24209**	24667	TL	TL	**9099**	11180	168	**0**	**-3.7**	-0.4	-0.4
A-84-220-20	25	**14852**	**14852**	**14852**	**14852**	**228**	322	**1135**	2911	1502	**3**	10.8	15.4	**6.1**
	50	**9082**	9016	**9082**	9673	**4599**	TL	**2623**	8131	1324	**0**	9.2	**6.5**	10.6
	75	7899	**7950**	**9031**	**9031**	TL	TL	**3**	8940	970	**0**	3.6	**-1.0**	1.8
A-84-220-103	25	**39718**	**39718**	**39718**	**39718**	**4536**	6833	**4519**	10673	12958	**1678**	**31.3**	51.8	51.7
	50	23545	**23560**	**26836**	**26836**	TL	TL	**90**	17750	433	**0**	**-6.5**	-3.5	-3.4
	75	**22463**	**22463**	**24794**	**24794**	TL	TL	**1**	4102	20	**0**	**-7.1**	-1.2	0.6
P-42-142-11	25	**6087**	**6087**	**6087**	**6087**	109	**100**	**626**	1455	2212	**0**	**0.0**	17.0	**0.0**
	50	**4551**	**4551**	**4551**	**4551**	**65**	308	**27**	3281	9	**0**	5.3	**0.0**	5.3
	75	**4512**	**4512**	**4512**	**4512**	**2**	**2**	**0**	**0**	**0**	**0**	**0.0**	**0.0**	**0.0**
P-42-142-65	25	**20791**	20702	**20791**	20838	**4306**	TL	**3455**	9796	14014	**1208**	**7.2**	9.5	9.5
	50	**13641**	13585	**14664**	15748	TL	TL	**4916**	15440	1675	**0**	-1.2	1.1	**-1.3**
	75	**13070**	13069	**13507**	13646	TL	TL	**7029**	10354	486	**0**	2.7	**-1.0**	**-1.0**
P-61-205-20	25	**7472**	**7472**	**7472**	**7472**	863	**826**	**1144**	3930	2879	**170**	4.4	**1.1**	1.7
	50	**5906**	5809	**6010**	**6010**	TL	TL	**18**	9627	1768	**0**	3.6	**0.0**	3.4
	75	**5677**	**5677**	**5677**	**5677**	3	**2**	**0**	**0**	**0**	**0**	6.3	**0.0**	6.1
P-61-205-98	25	**25984**	25934	**26080**	33256	TL	TL	**5434**	13316	10608	**0**	**14.9**	19.2	15.3
	50	**16646**	16597	**18500**	20187	TL	TL	**6308**	16560	584	**0**	**-1.3**	1.5	1.5
	75	**16423**	**16423**	**18362**	**18362**	TL	TL	**0**	5900	57	**0**	-2.3	**-4.4**	**-4.4**
P-123-350-33	25	**9397**	9312	**9397**	9451	**3112**	TL	**1828**	6525	2603	**483**	12.6	**11.6**	22.1
	50	**6254**	6253	**7211**	**7211**	TL	TL	**46**	8815	152	**0**	-0.4	**-3.0**	3.3
	75	**6085**	**6085**	**6133**	**6133**	TL	TL	**78**	980	3	**0**	12.6	**0.0**	13.0
P-123-350-170	25	31832	**31908**	**33016**	41409	TL	TL	**6189**	19077	1201	**0**	**12.9**	17.6	16.3
	50	20223	**20228**	**25475**	**25475**	TL	TL	**1**	1601	**0**	**0**	271.1	-0.5	**-2.0**
	75	**19489**	**19489**	**22070**	**22070**	TL	TL	**0**	**0**	**0**	**0**	372.1	-0.3	**-0.8**
V-60-218-20	25	**14282**	**14282**	**14282**	**14282**	**33**	92	**503**	1454	3	**0**	10.6	15.6	**6.0**
	50	**9358**	**9358**	**9358**	**9358**	**221**	338	**1902**	4633	25	**0**	**0.0**	10.2	**0.0**
	75	8389	**8446**	**9197**	**9197**	TL	TL	**29**	11093	1073	**0**	**-0.7**	0.0	1.1
V-60-218-103	25	**63825**	**63825**	**63825**	**63825**	**499**	2155	**4663**	9159	**847**	1023	-	-	-
	50	29720	**29774**	**30964**	37865	TL	TL	**8795**	20396	1166	**0**	**5.6**	11.7	11.5
	75	**24592**	**24592**	**29862**	**29862**	TL	TL	**0**	12096	82	**0**	**-0.7**	5.7	4.1
V-84-279-23	25	**12678**	**12678**	**12678**	**12678**	**487**	2353	**1347**	4449	1559	**0**	**52.2**	56.0	58.4
	50	7966	**8043**	**8505**	8780	TL	TL	**1782**	9186	1325	**0**	**-0.6**	**-0.6**	3.0
	75	**7254**	7215	**7254**	**7254**	**1920**	TL	**23**	9823	780	**0**	1.0	**0.0**	1.0
V-84-279-130	25	**71633**	71629	**72028**	72984	TL	TL	**5945**	11495	4430	**1608**	-	-	-
	50	**34076**	**34076**	**43672**	**43672**	TL	TL	**133**	10616	391	**0**	**-12.5**	-5.7	-9.5
	75	**26194**	**26194**	**35006**	**35006**	TL	TL	**0**	**0**	**0**	**0**	**-10.5**	-3.4	-7.5
V-146-401-38	25	**16670**	**16670**	**16670**	**16670**	2527	**1767**	**1450**	6116	1874	**41**	26.3	**4.7**	28.3
	50	10550	**10642**	**11896**	**11896**	TL	TL	**40**	12493	66	**0**	-2.1	**-3.3**	8.4
	75	8712	**9085**	**10167**	**10167**	TL	TL	**0**	4859	**0**	**0**	**-1.4**	-0.6	8.6
V-146-401-194	25	**76650**	76649	77667	**76742**	TL	TL	**6110**	20367	1972	**244**	**26.6**	42.5	42.9
	50	**39804**	39801	**48628**	**48628**	TL	TL	**0**	2100	**0**	**0**	477.8	**-0.7**	**-0.7**
	75	**31785**	**31785**	**40510**	**40510**	TL	TL	**0**	**0**	**0**	**0**	542.0	**-0.8**	**-0.8**

using the current LP solution as guideline. Unfortunately, this turned out to spend too much time without significantly improving the final results.

When comparing BC and BC+, we observe that the major difference is in the number of branching nodes:

Whereas both settings seem to achieve similar quality of solutions within the given time limit, BC+ stays at the root node in the majority of the cases, whereas BC creates larger branching trees. This clearly indicates a typical trade-off between branching and separation: separation of fractional points is computationally more demanding, but achieves better lower bounds and smaller branching trees, in general.

Finally, we point out that there are no infeasible instances in this set but with different parameters (e.g., for the numbers of allowed travel times) infeasibility due to the limited battery size might occur. Those cases can often be detected already early when building the energy-indexed graph: If a source node of some required arc cannot be reached with any feasible SOC, there is no feasible solution, cf. Section 3. On the other hand, when the battery limit is not restricting optimality can be proven by the initial RC heuristic: If the optimal DRPP solution is feasible for the eARP, the solution is optimal.

6.3 How precise is the energy-indexed model?

By constructing the energy-indexed graph, we discretize the SOC function and thus reduce the potential SOC levels. A very fine-grained SOC discretization could lead to longer computing times due to the expansion of the energy-indexed graph. On the other hand, a coarse-grained discretization together with the rounding-down of the SOC might lead to an overestimation of the optimal travel times. In this section we look for the answers to the following questions: 1) How much do we overestimate the travel times by working on the energy-indexed graph? 2) Do we sacrifice the quality of solutions by working on a coarse-grained discretization? and 3) What is the level of granularity that gives us a tractable mathematical model without sacrificing too much solution quality?

Recall that in our default experiments we round down the SOC after traversing an arc to the nearest $D = 1000$ kWs. This may allow us to partially account for the inherent uncertainty in the input data, but it may also lead to overestimated total travel times (the rounding effect) or missed optimal solutions (the discretization effect). Thus, in the following study, we consider three different discretization levels, $D \in \{500, 1000, 2000\}$ in kWs, on a small set of instances for which optimal solutions can be obtained with our exact method with a time limit of one day. An optimal route, i.e., a collection of m walks, $\mathcal{W}_1, \ldots, \mathcal{W}_m$, obtained with the discretization level D is only an approximation of the optimal route, so we re-evaluate each $\mathcal{W}_i$ using a more refined discretization level $D' \in \{1, 100, 500\}$ in two possible ways, according to the two encodings discussed in Section 3:

- Encoding as a sequence of traversed arcs, $R(D, D')$: We fix the walks $\mathcal{W}_1, \ldots, \mathcal{W}_m$ obtained with D and re-calculate the travel times with respect to the discretization level D'. To this end we apply the dynamic program from Theorem 3. Observe that the travel time obtained with D is always an upper bound for the travel time based on the more refined discretization level D'.
- Encoding as a sequence of served arcs, $S(D, D')$: From the walks $\mathcal{W}_1, \ldots, \mathcal{W}_m$ obtained with D, we derive the sequences of served required arcs $\mathcal{R}_1, \ldots, \mathcal{R}_m$ by assuming that the earliest visit of a required arc over all m routes defines its service (ties are broken by lower vehicle index). Then, we apply Algorithm 1 (with no limit on the number of labels) based on discretization level D' to obtain an optimal walk for each sequence and re-calculate the total travel time.

If $OPT(D)$ denotes the optimal travel time with respect to discretization level D, and $D' \leq D$ is a more refined discretization level, then we have

$$OPT(D) \geq R(D, D') \geq S(D, D') \geq OPT(D').$$

Table 2 compares the optimal travel times obtained with $D \in \{500, 1000, 2000\}$ with the re-evaluated travel times for $D' \in \{1, 100, 500\}$. We report the relative gaps of $OPT(D)$, $S(D, D')$, and $R(D, D')$ with respect to reference value $OPT(500)$, denoted by $OPT_g(D)$, $S_g(D, D')$, and $R_g(D, D')$, respectively, calculated as

$$OPT_g(D) = \frac{OPT(D) - OPT(500)}{OPT(500)},\ S_g(D, D') = \frac{S(D, D') - OPT(500)}{OPT(500)},\ R_g(D, D') = \frac{R(D, D') - OPT(500)}{OPT(500)}.$$

Table 2: Relative deviations of total travel time with respect to $OPT(500)$ for different discretization levels

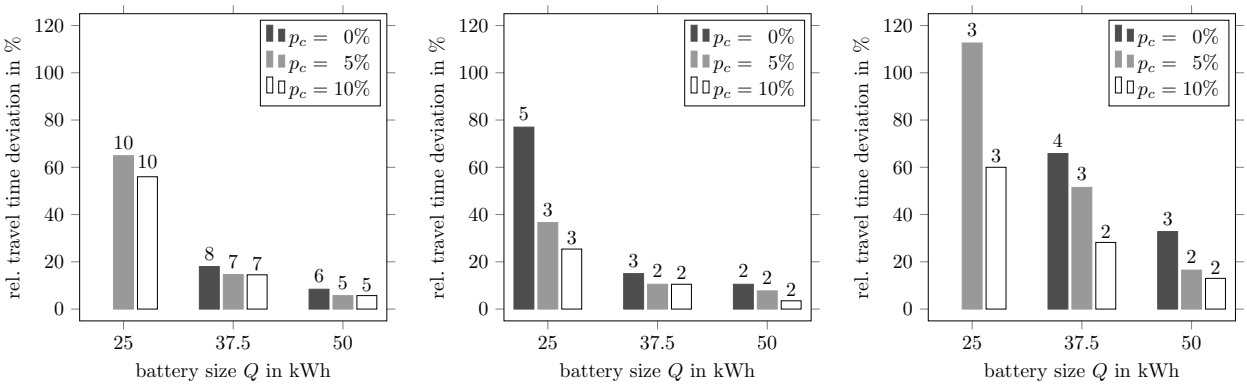
		D'	-	500		100		1	
instance	Q	D	OPT_g	S_g	R_g	S_g	R_g	R_g	$t[s]$
A-31-105-47	25	500	0.0	0.0	0.0	-2.3	-2.3	-2.7	298
		1000	2.9	0.0	0.0	-2.3	-2.3	-3.0	30
		2000	7.0	1.0	1.0	-1.2	-1.2	-1.8	23
	50	500	0.0	0.0	0.0	-0.2	-0.2	-0.2	17422
		1000	0.7	0.7	0.7	0.7	0.7	0.7	2229
		2000	0.7	0.7	0.7	0.7	0.7	0.7	224
A-58-168-15	25	500	0.0	0.0	0.0	-2.3	-2.3	-3.0	113
		1000	3.5	0.0	0.0	-2.3	-2.3	-3.0	10
		2000	11.6	0.4	0.9	-1.8	-1.4	-2.1	4
	50	500	0.0	0.0	0.0	0.0	0.0	0.0	11122
		1000	0.4	0.4	0.4	0.4	0.4	0.4	7138
		2000	0.4	0.4	0.4	0.4	0.4	0.4	122
A-84-220-20	25	500	0.0	0.0	0.0	-2.8	-2.8	-3.6	1633
		1000	4.1	0.0	0.0	-2.8	-2.8	-3.6	262
		2000	17.1	3.3	3.4	0.8	0.8	0.1	56
	50	500	0.0	0.0	0.0	-0.3	-0.3	-0.3	8264
		1000	2.6	0.0	0.0	-0.3	-0.3	-0.3	4632
		2000	12.6	12.6	12.6	12.6	12.6	12.6	1590
P-42-142-11	25	500	0.0	0.0	0.0	-3.5	-3.5	-3.9	44
		1000	5.0	0.0	0.0	-3.5	-3.5	-3.9	44
		2000	14.1	0.0	0.0	-3.5	-3.5	-3.9	15
	50	500	0.0	0.0	0.0	0.0	0.0	0.0	5
		1000	0.9	0.9	0.9	0.9	0.9	0.9	64
		2000	6.2	6.2	6.2	6.2	6.2	6.2	343
V-60-218-20	25	500	0.0	0.0	0.0	-3.3	-3.3	-4.0	373
		1000	4.2	0.2	0.2	-2.9	-2.9	-3.6	38
		2000	12.2	0.2	0.2	-2.9	-2.9	-3.6	15
	50	500	0.0	0.0	0.0	0.0	0.0	0.0	9890
		1000	0.5	0.2	0.2	0.2	0.2	0.2	854
		2000	6.7	0.0	0.0	0.0	0.0	0.0	436

In column $t[s]$ we provide the computing times for calculating $OPT(D)$. While the method for evaluating $R(D, D')$ is extremely fast for all instances and values D' (it requires less than one second), Algorithm 1 (which is applied with no upper limit on the number of labels) needed for the calculation of $S(D, D')$ can take a significant amount of time, cf. Section 6.2, which is why $S_g(D, 1)$ is left out from the table.

The values provided in column OPT_g indicate the relative increase of the calculated travel time when coarsening the level of discretization from 500, to 1000, respectively 2000. We observe that there is an overestimation of the total travel time caused by discretization: it is higher for the smaller battery ($Q = 25$) and ranges from 7% to 17%. However, for the larger battery ($Q = 50$) it is less pronounced and varies between 0.4% and 12.6%.

The next question is related to the robustness of the optimal solution found on the energy-indexed graph: When comparing the total travel times of two solutions obtained by $D = 500$ and $D = 1000$, we observe that optimality is not sacrificed when less granularity is applied, i.e., in most of the cases, the relative gaps S_g and R_g are the same. By moving towards a more coarse-grained discretization ($D = 2000$) we notice that the quality of the obtained solution deteriorates.

It is interesting to see that re-evaluating the travel times using method S rarely leads to better solutions than method R indicating that the walks usually do not change after applying Algorithm 1. Only for



(a) instance A-31-105-47 (b) instance P-61-205-20 (c) instance V-60-218-20

Figure 8: Solution comparison for different battery sizes and numbers of charging facilities.

instances A-58-168-15 and A-84-220-20, battery size $Q = 25$, and discretization level $D = 2000$, we can observe a slight improvement when using method S.

We conclude that using discretization level $D = 1000$ leads to reasonable runtimes and to negligible quality losses compared to $D = 500$. In case more accurate information on the actual energy consumption is available, one can efficiently re-evaluate the solutions with method $R(D, D')$ with very low values for D'.

6.4 What are the costs of "going green"?

There are multiple cost factors to be considered when a logistic company decides to "go green" by acquiring an EV fleet: 1) the acquisition costs of vehicles, 2) the costs for building (or using) charging infrastructure, 3) the service costs (i.e., the increase of total travel times due to the limited battery capacities and limited charging infrastructure), and 4) further operational costs (for running the fleet and maintaining the overall infrastructure). On the other hand, many companies are forced today to rethink their operations and to turn green, due to possible governmental regulations or threatened reputation when it comes to their environmental impact and ecological footprint.

A typical company in the logistics sector would therefore like to know what are the estimated costs associated with a replacement of the traditional combustion engine vehicle fleet with EVs. Clearly, a notable advantage of dynamic charging along the arcs versus stationary charging is the elimination of waiting times at the stations (either for charging the batteries or for waiting in the queues at the stations). Hence, the next interesting question is: Can the fleet acquisition costs be alleviated in the long-term if EVs would be equipped with smaller batteries provided that sufficient arcs with dynamic charging are available? To answer these questions, we analyze in detail the obtained optimal solutions for different battery sizes and relative amounts of dynamic charging arcs. Finally, we give insights and recommendations on strategic decisions related to the size of the vehicle fleet, the battery size, and the amount of charging facilities available.

Figure 8 compares optimal solutions for three instances, different battery sizes $Q \in \{25, 37.5, 50\}$, and different charging infrastructure, obtained with the default discretization level of 1000 kWs and re-evaluated with method R, see Section 6.3, based on a more fine-grained discretization of 1 kWs. We report the relative deviations $(R(1000, 1) - OPT_D)/OPT_D$ (in percent) from the optimal DRPP solution with travel time OPT_D, which can be interpreted as driving with combustion engine vehicles without range limit (resulting in a single route). These deviations are always non-negative, as the DRPP solution is naturally a lower bound for the eARP solution. We investigate three scenarios: There are no charging facilities at all (except at the depot), with wireless charging equipment installed on $p_c = 5\%$ of the arcs (restricted to the set of required arcs), and with the default setting of 10% charging arcs, see Section 6.1. The values above the bars in Figure 8 present the number of vehicles used in the obtained solution.

We start by studying the effect of the battery capacity on the increase of travel times, compared to the

fleet with combustion engines. As shown in Figure 8, too small batteries without arc charging infrastructure may not even result in feasible routes. However, with charging infrastructure available ($p_c = 10\%$), even with the smallest batteries the relative increase of the travel times is rather moderate and ranges between 25.4% and 60%. Moreover, these numbers significantly drop down by either increasing the battery capacity, improving the infrastructure, or both. In the most optimistic scenario we tested (namely, $Q = 50$ and with $p_c = 10\%$ of all arcs having recharging equipment), the increase of the total travel time is almost negligible, and ranges from 3.5% (instance P) to 13% (instance V). In comparison to vehicles with combustion engine we observe in most of the considered cases only minor increases of total travel times if the battery size is 50 kWh, even when only a few charging arcs are available.

Note that 50 kWh is far below the technically possible 630 kWh for the Eforce E18 truck. However, since the battery is a major part of the vehicle acquisition costs and there are environmental concerns in its production and recycling, reducing its capacity to a reasonable size is highly desired. Hence, the next question is, what is the acceptable trade-off between the increase of travel times and the investment in the fleet/infrastructure? If the decision-maker asks for a solution with a total travel time not more than a fixed percentage above the solution with combustion-engined vehicles, there are sometimes multiple options to reach this goal. For example, for instance V-60-218-20 we can obtain a 35% target by using 37.5 kWh batteries together with 10% charging arcs and two vehicles, or alternatively with 50 kWh batteries, no charging facilities, and three vehicles.

We observe that possible savings in travel times (compared to the nominal travel times obtained with the smallest battery and no charging infrastructure) are directly correlated with the number of required routes. Note that we do not limit the number of vehicles in our tests to see how optimal solutions are structured, which means that the size of the fleet is implicitly optimized in our models. So, if a larger battery or a better infrastructure allows to reduce the size of the fleet, then the savings can be very high. For example, the savings are as big as 60% (for instance P with $Q = 25$) when more charging arcs are available. Similarly, keeping the same infrastructure (e.g., $p_c = 10\%$) and using a slightly larger battery ($Q = 37.5$) can lead to savings of up to 75% of the travel time (cf. instance A).

All these results lead us to the conclusion that decision-makers need to carefully examine the trade-offs between the various cost factors listed above. High capital expenditures necessary for building the charging infrastructure can reduce vehicle acquisition costs (and the further operational expenditures related to the maintenance of the fleet). In the long term, a fleet with smaller battery sizes combined with appropriate charging infrastructure could offer a more cost efficient solution, without significantly sacrificing the service costs (i.e., with the relative increase of travel times remaining below a desired target level).

7 Conclusions

In this article we proposed and studied an arc routing problem with a fleet of electric vehicles, where charging of vehicles is possible at the depot or along some dedicated arcs of the network. Non-linear charging functions and multiple options for traversing arcs (considering different speeds or charging options) are discretized and modeled in an energy-indexed graph. An exact branch-and-cut framework has been developed to solve challenging and realistic instances to (near-)optimality. This framework is built starting from a new ILP formulation and some families of valid inequalities. Two possible ways for encoding feasible solutions are proposed, and the problems of verifying their feasibility and finding the optimal decoding are analyzed from the computational complexity perspective. For efficient initialization of upper bounds, three heuristics that exploit the solution encoding techniques have been integrated in this framework.

Our computational results have shown that this new exact solution framework based on the concept of energy-indexed graphs is capable of solving sparse instances with hundreds of arcs to optimality. Gaps between lower and upper bounds were significantly reduced thanks to the tight LP bounds of the mathematical formulation and the integration of heuristics. The computational efficiency of the latter ones was improved by decoding and solution tracking mechanisms.

Concerning future work, for solving even larger instances close to optimality, more complex heuristic moves together with diversification mechanisms to escape local optima might be necessary. One has to keep

in mind, however, that more sophisticated metaheuristics or matheuristics probably induce higher efforts in sequence decoding potentially leading to impracticable runtimes.

Finally, our managerial findings show that there is a trade-off between fleet acquisition costs, investment costs for the charging infrastructure, and the potential increase of travel times due to limited battery range. These factors have to be carefully measured by the decision makers, to achieve cost efficient solutions in the long run. These findings naturally lead to other interesting optimization problems related to strategic decisions for going green in the logistics sector. Relevant questions to be studied in the future include the network design of charging infrastructure or simultaneous optimization of the fleet size, dimensioning of the batteries and the charging infrastructure.

Acknowledgements

E. Fernandez was partially supported by the Spanish Ministry of Economy and Competitiveness through MINECO/FEDER grant MTM2015-63779-R. M. Leitner, I. Ljubić and M. Ruthmair were partially funded by the Vienna Science and Technology Fund through project ICT15-014.

References

[1] eRoadArlanda, 2017. URL `https://eroadarlanda.com/`. accessed on 2019-06-19.

[2] J. Asamer, A. Graser, B. Heilmann, and M. Ruthmair. Sensitivity analysis for energy demand estimation of electric vehicles. *Transportation Research Part D: Transport and Environment*, 46:182–199, 2016.

[3] T. Ávila, A. Corberán, I. Plana, and J. M. Sanchis. A new branch-and-cut algorithm for the generalized directed rural postman problem. *Transportation Science*, 50(2):750–761, 2016.

[4] E. Bartolini, J.-F. Cordeau, and G. Laporte. An exact algorithm for the capacitated arc routing problem with deadheading demand. *Operations Research*, 61(2):315–327, 2013.

[5] J. E. Beasley. Route first—cluster second methods for vehicle routing. *Omega*, 11(4):403–408, 1983.

[6] T. Bektas and G. Laporte. The Pollution-Routing Problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011.

[7] J. M. Belenguer, E. Benavent, and S. Irnich. The capacitated arc routing problem: Exact algorithms. In *Arc Routing*, pages 183–222. Springer, 2000.

[8] Z. Bi, T. Kan, C. C. Mi, Y. Zhang, Z. Zhao, and G. A. Keoleian. A review of wireless power transfer for electric vehicles: Prospects to enhance sustainable mobility. *Applied Energy*, 179:413–425, 2016.

[9] C. Bode and S. Irnich. Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, 60(5):1167–1182, 2012.

[10] Z. Chen, F. He, and Y. Yin. Optimal deployment of charging lanes for electric vehicles in transportation networks. *Transportation Research Part B: Methodological*, 91:344–365, 2016.

[11] N. Christofides, V. Campos, A. Corberán, and E. Mota. An algorithm for the rural postman problem on a directed graph. In *Netflow at Pisa*, pages 155–166. Springer, 1986.

[12] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[13] Á. Corberán and G. Laporte, editors. *Arc routing: problems, methods, and applications*, volume 20. SIAM, 2014.

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

[15] Eforce. Eforce EF18, 2018. URL `https://www.eforce.ch/products/ef18`. accessed on 2018-11-20.

[16] G. Ferro, M. Paolucci, and M. Robba. An optimization model for electrical vehicles routing with time of use energy pricing and partial recharging. *IFAC-PapersOnLine*, 51(9):212–217, 2018.

[17] M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229, 2017.

[18] A. Franceschetti, D. Honhon, T. Van Woensel, T. Bektaş, and G. Laporte. The time-dependent pollution-routing problem. *Transportation Research Part B: Methodological*, 56:265–293, 2013.

[19] R. Fukasawa, Q. He, F. Santos, and Y. Song. A joint vehicle routing and speed optimization problem. *INFORMS Journal on Computing*, 30(4):694–709, 2018.

[20] B. L. Golden, J. S. DeArmon, and E. K. Baker. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, 10(1):47–59, 1983.

[21] L. Gouveia, M. Leitner, and M. Ruthmair. Layered graph approaches for combinatorial optimization problems. *Computers & Operations Research*, 102:22–38, 2019.

[22] M. H. Hà, N. Bostel, A. Langevin, and L.-M. Rousseau. Solving the close-enough arc routing problem. *Networks*, 63(1):107–118, 2014.

[23] IEA. Global EV outlook 2018, 2018. URL `https://www.iea.org/gevo2018/`. accessed on 2019-06-18.

[24] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In *Column generation*, pages 33–65. Springer, 2005.

[25] P. Lacomme, C. Prins, and W. Ramdane-Cherif. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(1-4):159–185, 2004.

[26] J. K. Lenstra and A. R. Kan. On general routing problems. *Networks*, 6(3):273–280, 1976.

[27] R. Linke. The real barriers to electric vehicle adoption. *MIT Sloan Review*, 2017.

[28] L. Lozano and A. L. Medaglia. On an exact method for the constrained shortest path problem. *Computers & Operations Research*, 40(1):378–384, 2013.

[29] S. Lukić and Ž. Pantić. Cutting the Cord: Static and Dynamic Inductive Wireless Charging of Electric Vehicles. *IEEE Electrification Magazine*, 1(1):57–64, 2013.

[30] O. Lum, B. Golden, and E. Wasil. An open-source desktop application for generating arc-routing benchmark instances. *INFORMS Journal on Computing*, 30(2):361–370, 2018.

[31] G. Macrina, L. D. P. Pugliese, F. Guerriero, and G. Laporte. The green mixed fleet vehicle routing problem with partial battery recharging and time windows. *Computers & Operations Research*, 101: 183–199, 2019.

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

[33] C. Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974.

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

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

[36] J. Qian and R. Eglese. Fuel emissions optimization in vehicle routing problems with time-varying speeds. *European Journal of Operational Research*, 248(3):840–848, 2016.

[37] G. R. Raidl and B. Hu. Enhancing genetic algorithms by a trie-based complete solution archive. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 239–251. Springer, 2010.

[38] M. Ruthmair and G. R. Raidl. A memetic algorithm and a solution archive for the rooted delay-constrained minimum spanning tree problem. In R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, editors, *Proceedings of the 13th International Conference on Computer Aided Systems Theory: Part I*, volume 6927 of *LNCS*, pages 351–358. Springer, 2012.

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

[40] R. Shuttleworth, B. L. Golden, S. Smith, and E. Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 487–501. Springer, 2008.

[41] G. Ulusoy. The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22(3):329–337, 1985.

[42] T. Vidal. Node, edge, arc routing and turn penalties: Multiple problems—one neighborhood extension. *Operations Research*, 65(4):992–1010, 2017.

[43] T. Zündorf. Electric vehicle routing with realistic recharging models. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, Germany, 2014.

A Appendix

A.1 Heuristic results

In Table 3 we compare the algorithmic performance of our heuristic methods for the eARP from Section 4. The three construction heuristics SM, RC, and CM, are applied to the city instances with a time limit of one hour and a memory limit of 8 GB. Note that the local search phase is always included in the presented results. The quality of some heuristic solution is measured as the relative travel time deviation to the best solution found by the BC approaches in Section 6.2. Dashes "-" denote the cases where no feasible solution has been found throughout the search process within the time limit.

Additionally, we measure the speedup factor obtained when using the solution archive by dividing the running time t_{noA} without archive by the running time t_{A} when using it. If t_{noA} reaches the time limit of one hour the corresponding speedup value is a lower bound, while we use a dash "-" to denote the cases where t_{A} exceeds the time limit. Note that as soon as 90% of the memory limit is occupied no further sequences are stored in the archive to ensure the memory requirements.

Table 3: Performance of heuristics (best values per line are marked **bold**)

			relative primal bound in %												runtime in seconds												archive speedup $t_{\text{noA}}/t_{\text{A}}$											
				SM			RC				CM				SM				RC				CM				SM				RC				CM			
instance	C	Q	$K=1$	5	20	∞	1	5	20	∞	1	5	20	∞	1	5	20	∞	1	5	20	∞	1	5	20	∞	1	5	20	∞	1	5	20	∞	1	5	20	∞
A-31-105-8	5	25	-	1.9	1.4	**0.0**	-	1.9	1.4	**0.0**	-	1.9	1.8	**0.0**	**0**	**0**	**0**	2	**0**	**0**	**0**	1	**0**	**0**	**0**	1	1.0	**3.8**	2.2	1.7	1.0	3.2	2.9	2.2	1.0	2.9	2.1	1.7
		50	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0**	1	5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2	2.0	**2.1**	1.9	1.3	1.0	1.0	1.0	0.9	1.5	1.7	1.8	1.9
		75	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0**	1	4	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2	1.7	**1.9**	**1.9**	1.3	1.0	0.5	1.0	1.1	1.0	1.7	1.7	1.1
A-31-105-47	2	25	-	-	4.6	**2.8**	-	-	9.9	6.4	-	-	8.2	7.8	**0**	**0**	13	674	**0**	**0**	11	861	**0**	**0**	12	897	1.0	1.0	5.4	2.1	2.0	1.7	9.1	4.2	2.0	1.7	**10.0**	4.0
		50	8.4	8.4	7.1	7.1	6.1	7.6	2.2	2.2	6.1	7.6	**1.1**	**1.1**	**1**	7	38	1134	**1**	4	34	2044	**1**	5	34	1920	2.5	3.1	3.1	1.7	5.9	4.9	5.2	1.8	5.0	6.1	**6.3**	1.9
		75	9.2	8.5	8.6	7.6	8.9	8.9	**2.1**	**2.1**	6.8	6.8	**2.1**	**2.1**	2	10	70	1210	**1**	9	59	3382	2	14	78	3278	2.7	2.6	2.5	1.8	3.9	**4.0**	3.4	1.1	3.2	3.3	3.1	1.1
A-58-168-15	7	25	-	-	**0.0**	**0.0**	-	-	**0.0**	2.7	-	-	1.2	0.8	**0**	**0**	2	36	**0**	**0**	1	33	**0**	**0**	1	19	1.0	1.0	3.2	1.4	2.0	1.5	3.3	2.2	1.0	1.5	**5.2**	2.5
		50	**0.0**	**0.0**	**0.0**	**0.0**	11.6	10.7	0.5	0.5	**0.0**	**0.0**	**0.0**	**0.0**	**0**	2	11	82	**0**	1	9	106	**0**	2	11	60	3.5	**3.6**	3.1	2.0	2.1	2.0	3.1	2.2	3.3	3.5	3.2	3.5
		75	11.0	11.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1	5	13	74	**0**	**0**	**0**	9	1	5	23	114	2.3	2.2	**3.2**	1.9	1.0	1.0	1.0	1.1	2.9	3.0	2.9	2.8
A-58-168-81	1	25	-	-	**3.2**	149.8	-	-	9.6	180.7	-	-	9.6	-	**0**	**0**	173	TL	**0**	**0**	123	TL	**0**	**0**	125	TL	1.0	1.0	**7.0**	-	1.6	2.0	6.9	-	1.6	2.0	6.2	-
		50	**4.2**	4.4	6.0	303.7	8.9	8.9	5.1	232.5	8.9	8.9	5.1	-	7	45	257	TL	**4**	25	278	TL	**4**	25	302	TL	3.0	5.0	4.1	-	7.0	7.1	7.0	-	7.1	**7.2**	6.9	-
		75	1.5	-0.9	**-3.7**	332.7	-0.4	-0.4	-0.4	185.3	-0.4	-0.4	-0.4	-	14	82	349	TL	**10**	70	452	TL	**10**	70	461	TL	4.7	3.5	2.6	-	**5.3**	5.2	4.9	-	**5.3**	5.2	4.7	-
A-84-220-20	11	25	-	-	10.8	4.1	-	-	15.4	8.2	-	-	6.1	**1.5**	**0**	**0**	10	202	**0**	**0**	7	143	**0**	**0**	9	134	1.0	1.0	6.1	1.7	1.7	2.1	**8.2**	2.1	0.8	1.0	5.5	1.9
		50	9.8	9.8	9.2	9.8	14.2	14.2	6.5	**6.3**	18.9	17.8	10.6	10.6	**1**	7	45	393	3	5	18	238	**1**	5	30	360	2.9	3.3	3.1	2.2	1.1	**3.8**	2.2	1.7	3.2	3.7	3.0	2.0
		75	3.6	3.6	3.6	3.6	-1.0	-1.0	-1.0	**-2.3**	1.8	1.8	1.8	1.8	**2**	19	90	704	**2**	12	69	349	**2**	16	93	642	3.0	3.0	**3.5**	2.0	2.6	2.6	2.6	1.7	2.4	2.5	2.2	2.0
A-84-220-103	3	25	-	-	**31.3**	215.7	-	-	51.8	186.1	-	-	51.7	-	**0**	**0**	358	TL	**0**	1	201	TL	**0**	1	171	TL	1.0	1.0	5.5	-	2.0	2.5	**14.9**	-	2.3	2.5	12.6	-
		50	-5.5	-5.7	**-6.5**	498.4	-0.2	-0.2	-3.5	248.2	3.6	3.6	-3.4	-	22	135	774	TL	**8**	55	376	TL	**8**	53	427	TL	4.3	4.2	4.3	-	6.1	5.8	5.2	-	**7.6**	**7.6**	6.1	-
		75	-6.9	**-8.3**	-7.1	590.6	-1.2	-1.2	-1.2	221.0	2.9	2.9	0.6	-	39	234	1307	TL	**13**	98	668	TL	21	159	1318	TL	3.7	2.9	2.4	-	3.0	2.9	2.9	-	**4.6**	4.4	2.7	-
P-42-142-11	9	25	-	-	**0.0**	9.5	-	-	17.0	10.7	-	-	**0.0**	9.5	**0**	**0**	1	3	**0**	**0**	1	3	1	**0**	1	4	1.0	1.0	2.7	1.8	1.0	1.2	**3.0**	2.8	0.1	0.9	2.8	1.5
		50	5.3	5.3	5.3	5.3	**0.0**	**0.0**	**0.0**	**0.0**	5.3	5.3	5.3	5.3	**0**	1	5	12	**0**	**0**	1	5	**0**	1	4	10	2.5	2.5	2.5	2.3	2.0	2.4	**2.8**	2.4	2.0	2.4	2.6	2.4
		75	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	1	4	11	**0**	**0**	**0**	**0**	**0**	1	4	10	2.6	2.8	2.9	2.3	0.7	1.3	0.9	0.9	2.0	2.7	**3.0**	2.3
P-42-142-65	1	25	-	-	**7.2**	**7.2**	-	-	9.5	10.3	-	-	9.5	10.3	**0**	**0**	68	1420	**0**	**0**	68	2571	**0**	**0**	76	2813	1.0	1.0	3.0	2.0	2.0	2.3	**10.3**	1.4	2.2	2.3	9.8	1.3
		50	-2.0	0.4	-1.2	**-3.6**	1.4	1.1	1.1	1.1	-0.7	-0.1	-1.3	-1.3	4	24	148	2521	**3**	22	122	3580	**3**	17	130	2647	2.6	2.7	2.9	1.4	6.8	6.1	**7.4**	1.0	6.3	4.7	6.5	1.4
		75	2.7	2.7	2.7	3.5	0.4	**-1.0**	**-1.0**	**-1.0**	0.4	**-1.0**	**-1.0**	**-1.0**	8	38	264	TL	**3**	21	132	TL	**3**	21	132	TL	2.7	3.3	2.3	-	3.8	4.2	**4.7**	-	3.7	4.2	4.5	-
P-61-205-20	13	25	-	7.1	4.4	4.4	18.8	9.5	1.1	0.9	-	10.8	1.7	**0.6**	**0**	2	15	118	**0**	1	14	113	**0**	2	20	99	1.0	2.2	1.8	1.3	5.5	5.4	**5.8**	2.7	1.0	3.6	4.8	2.9
		50	3.6	3.6	3.6	3.6	6.0	0.4	**0.0**	**0.0**	3.4	3.4	3.4	3.4	**1**	7	37	245	**1**	5	12	110	**1**	5	26	117	2.8	2.9	3.1	1.4	**3.4**	**3.4**	3.3	2.7	2.6	3.2	3.1	2.8
		75	6.3	6.3	6.3	6.3	**0.0**	**0.0**	**0.0**	**0.0**	6.1	6.1	6.1	6.1	1	8	40	247	**0**	**0**	**0**	7	1	5	23	147	2.1	2.2	1.8	1.4	1.0	1.0	1.1	1.1	2.4	**3.0**	**3.0**	1.4
P-61-205-98	1	25	-	-	**14.9**	93.7	-	-	19.2	217.1	-	-	15.3	-	**0**	**0**	407	TL	**0**	1	260	TL	**0**	1	358	TL	1.0	1.0	7.1	-	2.9	3.3	**11.4**	-	3.0	3.1	9.9	-
		50	1.0	-0.2	**-1.3**	321.3	3.4	3.2	1.5	390.2	3.3	3.3	1.5	-	29	206	1455	TL	**18**	142	1068	TL	19	139	1195	TL	3.9	4.9	2.5	-	**8.0**	7.4	3.4	-	7.5	7.1	3.0	-
		75	-2.1	-1.7	-2.3	300.4	**-4.9**	**-4.9**	-4.4	269.6	**-4.9**	**-4.9**	-4.4	-	**64**	373	2399	TL	75	589	TL	TL	75	605	TL	TL	4.1	3.6	1.5	-	4.6	4.3	-	-	**4.7**	4.2	-	-
P-123-350-33	20	25	-	43.0	12.6	**2.1**	-	43.6	11.6	3.3	-	43.0	22.1	10.7	**0**	10	134	2030	**0**	8	131	1705	**0**	9	141	2799	1.0	2.6	2.6	1.4	1.2	**9.2**	5.0	2.1	1.1	3.7	3.9	1.3
		50	1.1	1.1	-0.4	-0.4	**-6.6**	**-6.6**	-3.0	-3.5	1.4	-1.4	3.3	3.3	**8**	53	445	TL	9	61	314	3257	**8**	92	785	TL	2.7	2.9	2.9	-	5.5	**5.6**	2.8	1.1	3.3	3.4	1.4	-
		75	12.6	12.6	12.6	12.6	3.7	**0.0**	**0.0**	**0.0**	13.0	13.0	13.0	15.2	14	97	498	TL	**10**	65	248	2629	18	126	1099	TL	1.7	1.8	1.7	-	2.2	2.3	**2.9**	1.4	2.7	2.8	1.7	-
P-123-350-170	2	25	-	-	**12.9**	553.7	-	-	17.6	-	-	-	16.3	-	**0**	**0**	TL	TL	1	8	TL	TL	1	7	3585	TL	1.0	1.0	-	-	3.6	**4.1**	-	-	3.6	**4.1**	1.0	-
		50	**-13.3**	-12.3	271.1	733.3	-3.3	-3.0	-0.5	-	-3.8	-3.8	-2.0	-	282	1925	TL	TL	181	1407	TL	TL	**131**	1038	TL	TL	6.9	1.9	-	-	9.4	2.6	-	-	**11.9**	3.5	-	-
		75	**-5.2**	-5.0	372.1	861.8	-0.4	-1.7	-0.3	-	-1.2	-2.9	-0.8	-	**643**	TL	TL	TL	849	TL	TL	TL	842	TL	TL	TL	**4.9**	-	-	-	4.2	-	-	-	4.3	-	-	-
V-60-218-20	8	25	-	-	10.6	**0.0**	-	-	15.6	**0.0**	-	-	6.0	**0.0**	**0**	**0**	8	175	**0**	**0**	7	121	**0**	**0**	9	144	1.0	1.0	2.9	2.2	1.0	1.3	4.6	3.4	1.0	1.0	4.9	**5.1**
		50	10.3	**0.0**	**0.0**	**0.0**	1.7	10.2	10.2	10.2	16.7	**0.0**	**0.0**	**0.0**	1	6	33	506	**0**	3	19	309	1	6	37	364	2.6	2.8	2.7	1.4	**7.0**	4.2	4.0	2.8	2.9	4.4	3.9	3.8
		75	-0.7	-0.7	-0.7	-0.7	-0.7	-0.7	0.0	**-1.8**	-0.6	1.1	1.1	1.1	2	12	64	604	**1**	6	19	212	3	9	73	335	2.3	2.2	2.1	1.7	**4.2**	**4.2**	2.7	2.7	2.9	2.6	1.6	1.6
V-60-218-103	1	25	-	-	-	**18.8**	-	-	-	297.1	-	-	-	-	**0**	**0**	**0**	TL	**0**	1	6	TL	**0**	1	6	TL	1.0	1.0	1.0	-	2.0	2.5	**4.3**	-	2.0	2.5	**4.3**	-
		50	7.4	6.6	**5.6**	355.2	20.4	19.7	11.7	791.1	20.4	16.0	11.5	-	18	111	602	TL	**13**	71	412	TL	**13**	54	433	TL	4.2	5.0	4.1	-	11.3	8.5	8.7	-	**11.6**	9.3	8.3	-
		75	**-0.7**	0.2	**-0.7**	488.5	8.6	5.7	5.7	798.6	8.6	4.1	4.1	-	31	217	1050	TL	**24**	227	1306	TL	25	216	2077	TL	5.4	4.9	3.4	-	7.0	**7.2**	2.8	-	6.7	6.3	1.7	-
V-84-279-23	14	25	-	-	52.2	12.4	-	-	56.0	**2.0**	-	-	58.4	5.3	**0**	**0**	13	280	**0**	**0**	17	216	**0**	**0**	20	411	1.0	1.0	3.0	1.6	1.4	1.9	**6.7**	3.7	1.2	1.0	4.8	3.2
		50	15.0	3.0	**-0.6**	**-0.6**	1.4	1.4	**-0.6**	**-0.6**	19.0	6.7	3.0	3.0	2	15	81	846	**1**	9	70	712	2	13	75	662	2.8	**4.3**	2.8	1.5	4.2	4.2	3.5	3.4	3.0	3.9	2.8	2.6
		75	3.0	**0.0**	1.0	1.0	3.0	**0.0**	**0.0**	**0.0**	3.0	1.0	1.0	1.0	3	29	77	760	**1**	7	33	336	2	11	72	356	**5.9**	3.4	4.0	2.0	3.2	3.4	3.0	4.3	4.6	5.6	3.7	2.4
V-84-279-130	3	25	-	-	-	337.3	-	-	-	**71.9**	-	-	-	-	**0**	**0**	**0**	TL	**0**	1	8	TL	**0**	1	7	TL	1.0	1.0	1.0	-	2.1	2.1	**2.5**	-	1.9	2.1	2.4	-
		50	-11.0	**-15.1**	-12.5	465.0	7.0	-2.9	-5.7	133.5	-0.1	-0.3	-9.5	-	46	311	2459	TL	**25**	204	1236	TL	29	197	1565	TL	3.4	6.1	1.5	-	13.5	13.5	2.9	-	**15.4**	12.4	2.3	-
		75	-9.9	-8.3	**-10.5**	604.9	-3.1	-4.4	-3.4	153.7	-8.0	-8.0	-7.5	-	104	661	TL	TL	**67**	581	TL	TL	82	643	TL	TL	4.8	4.6	-	-	8.3	6.2	-	-	**11.1**	5.6	-	-
V-146-401-38	21	25	-	-	26.3	109.1	-	-	4.7	**2.3**	-	-	28.3	31.1	**0**	**0**	496	TL	**0**	1	250	TL	**0**	**0**	876	TL	1.0	1.0	2.6	-	1.5	2.0	**4.4**	-	1.0	1.1	2.3	-
		50	-2.1	-2.1	-2.1	264.5	**-3.3**	**-3.3**	**-3.3**	-1.6	9.7	9.7	8.4	18.2	**14**	88	576	TL	**14**	106	1038	TL	15	98	755	TL	3.4	4.0	2.9	-	5.2	5.0	3.5	-	**6.1**	**6.1**	4.6	-
		75	-1.4	-1.4	-1.4	391.4	**-3.0**	**-3.0**	-0.6	0.0	8.6	8.6	8.6	134.6	41	294	1749	TL	36	278	816	TL	**34**	258	2302	TL	3.9	3.8	2.1	-	3.4	3.2	**4.1**	-	3.8	3.6	1.6	-
V-146-401-194	1	25	-	-	**26.6**	412.5	-	-	42.5	254.2	-	-	42.9	-	**0**	**0**	TL	TL	1	6	TL	TL	1	6	TL	TL	1.0	1.0	-	-	2.3	**3.3**	-	-	2.4	3.1	-	-
		50	-8.6	**-9.8**	477.8	586.6	0.6	-1.5	-0.7	-	0.2	-1.5	-0.7	-	320	1967	TL	TL	228	1188	TL	TL	**226**	1179	TL	TL	6.3	1.8	-	-	15.8	3.0	-	-	**15.9**	3.1	-	-
		75	**-8.6**	-7.5	542.0	724.2	-2.1	-2.1	-0.8	-	-2.1	-1.8	-0.8	-	811	TL	TL	TL	**393**	3080	TL	TL	394	TL	TL	TL	4.4	-	-	-	**9.2**	1.2	-	-	9.1	-	-	-