Arc routing with electric vehicles: dynamic charging and speed-dependent energy consumption

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

July 17, 2020
(previous version: 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. A computational study is performed on realistic problem instances. Besides analyzing the performance of all proposed methods, the obtained results also provide insights into strategic decisions related to the battery size and the amount of charging facilities.

Keywords: integer programming, 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) [33]. 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) [4, 34].

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 (`www.iea.org/gevo2018`, 2019-06-18). Nevertheless, EVs still make up less than 1% of passenger vehicles worldwide. 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.

Adoption of dynamic charging in production, manufacturing, logistics and seaport operations is already happening at a fast pace. Modern warehouses are nowadays largely automated by robots that have an electric drive and require frequent recharging. Dynamic charging can be implemented via the inductive power transfer (IPT) technology, which enables dynamic *wireless* charging [27]. In this scenario, a mobile electric device is recharged while moving along a dedicated lane equipped with an IPT system. The company [42], for example, offers automated wireless charging systems for forklifts, mobile robots and industrial trucks. According to [42], up to 30% higher fleet availability can be achieved by eliminating interruptions caused by stationary charging. Dynamic charging has also been already deployed for passenger and cargo transportation. Multiple bus lines in London, Turin and Madrid have been successfully equipped by IPT systems in a pilot study conducted by the company [23]. Recent surveys regarding wireless power transfer technologies for EVs can be found in Ahmad et al. [1], Bi et al. [11]. An alternative dynamic charging 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, see the pioneering project *eRoadArlanda* at the Stockholm airport Arlanda (`eroadarlanda.com`, 2019-06-19).

The utilization of inspection robots (especially in some complex and dangerous working environments) is an emerging and highly relevant application of arc routing. Inspection robots are typically equipped with visible-light cameras and/or infrared thermographs, and are sending the acquired information in realtime to the data center for further processing (see, e.g., Wang et al. [41] for recent applications arising in the maintenance of electric power systems, where a magnetic guidance system is used to guide the robots). Dynamic charging is expected to significantly improve the efficiency of inspection robots in particular when employed for daily maintenance tasks (see, e.g., examples of brachiating robots in Menéndez et al. [30] and the survey by Allan and Beaudry [2]).

Overview and scientific contribution. This article provides a first study on the use of EVs in the context of arc routing with possible dynamic charging. Given a 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 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 are:

- We introduce the concept of the *energy-indexed graph* that provides a generic way to deal with (i) 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 for EVs, while the third one has been addressed for node-routing problems, e.g., by Montoya et al. [31], Baum et al. [7], Froger et al. [19].
- We derive a mathematical model on the energy-indexed graph and develop an exact solution framework based on branch-and-cut.
- We propose to encode eARP solutions as sequences of required arcs, study the complexity of the underlying feasibility problems, and propose a labeling algorithm for solution decoding.
- Based on the latter result, we derive efficient heuristics for obtaining feasible routes, which 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.
- We conduct a computational study on realistic problem instances and give insights on strategic decisions related to 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. Section 3 studies heuristics together with a solution representation and a corresponding decoding procedure, whereas the algorithmic details of our branch-and-cut implementations are given in Section 4. A computational study and managerial insights are given in Section 5, and final conclusions are drawn in Section 6.

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. [33]. 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. [34]. 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 [9] 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. [20] 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 [35], 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 speed 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. [29] 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.

To deal with the problem of deciding on charging stops and the corresponding amount of re-charged energy between two services, Andelmin and Bartolini [3] and Froger et al. [19] enumerate all non-dominated paths between two customers and use these paths in their integer linear programming formulations. These path concepts are somehow related to our solution representation described in Section 3.1 when serving arcs instead of nodes and considering speed-dependent energy consumption and dynamic charging.

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 [15, 38] 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. [34] address this issue and discuss tractable models for transportation problems that will allow estimate charging and discharging behavior of EV batteries. Montoya et al. [31], Baum et al. [7], Froger et al. [19] 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. The authors assume concave charging functions since no (relevant amount of) energy is consumed during charging at a station.

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 [12, 14]. In particular, the eARP is equivalent to the well-known *Directed Rural Postman problem (DRPP)* [32] in case the battery capacity is not restrictive. In Section 2 we give an alternative definition of the eARP that also relates it to the *generalized directed rural postman problem (GDRPP)* introduced by Ávila et al. [5]. There are some similarities between the eARP and the capacitated ARP with deadheading demand [6] 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 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 $\mathcal{W} = (a^1, \dots, a^k)$ with associated travel times $\mathcal{T} = (t^1, \dots, t^k)$, $t^j \in T(a^j)$, $a^j \in A$, for all $1 \leq j \leq k$, is feasible if the following holds: (i) $\mathcal{W}$ starts and ends at the depot, and (ii) the SOC b^j after traversing arc $a^j \in \mathcal{W}$ must be in the interval $[0, Q]$ for all $0 \leq j \leq k$. SOC b^j after traversing arc a^j with travel time t^j is defined as $b^j = \beta(a^j, t^j, b^{j-1})$, $1 \leq j \leq k$, 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 in time t^j. The objective is to identify a set of m feasible walks $S = \{(\mathcal{W}^\ell, \mathcal{T}^\ell)\}_{\ell=1}^m$ that covers all the required arcs with minimum total travel time. The eARP is NP-hard since it boils down to the DRPP [25] in case the battery capacity is sufficiently large.

Assumptions. We next discuss four main assumptions made in the above definition of eARP and its solutions: (i) *Directed demand:* As opposed to several other arc routing problems eARP is based on a directed graph and assumes that demand is placed on arcs rather than on (undirected) edges. Extensions to the most general case of mixed (i.e., undirected and directed) demands are, however, straightforward for all developments proposed in this article. While we explicitly comment on this aspect for the mathematical model in Section 2, we refrain (for the sake of readability) from introducing the necessary additional notation and case distinctions in all other sections. (ii) *Finite sets of travel times and SOC values:* We 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 obtain at any node $u \in V$ and assume that B is finite, too. The latter assumption is w.l.o.g. 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 Pelletier et al. [34], where discretization is presented as an alternative for modeling the behavior of a battery during discharging and charging. (iii) *Speed selection:* While allowing a vehicle (driver) to select its speed (from a predefined set of speeds) seems reasonable in some applications (e.g., for inspection robots performing maintenance tasks), this may not be easily possible in others (e.g., vehicles traveling on roads since they depend on traffic conditions). To this end, it is crucial that all possible travel times on an arc correspond to speeds that are possible under typical conditions. Besides, we observe that driving slower than originally planned will (usually) induce a smaller energy consumption. While a different route may be preferable if one would have known the arising conditions beforehand, the planned route will nevertheless remain battery feasible and therefore not lead to severe problems (such as stranded vehicles

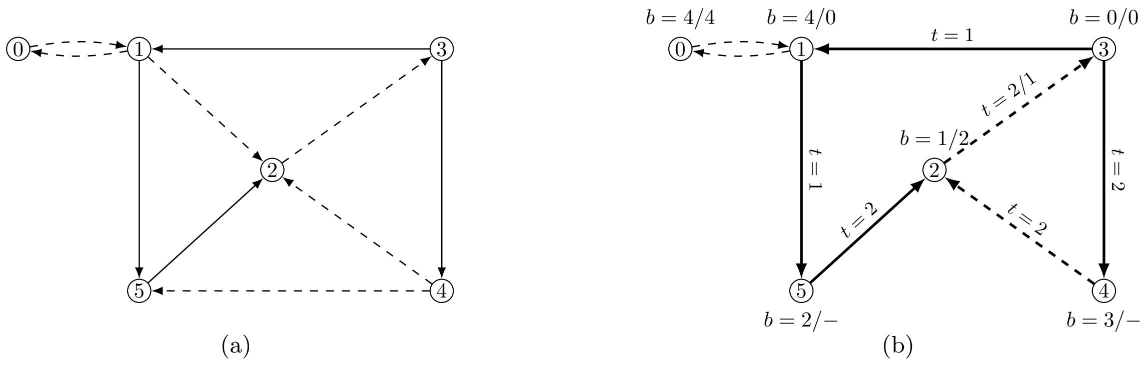


Figure 1: (a) Example instance; (b) optimal solution to it. Required arcs shown by solid lines, SOC values for the first and (where applicable) second visit given next to the nodes; used travel times next to the arcs.

with empty battery). (iv) *Recharging cycles:* eARP solutions may contain cycles without required arcs that are driven to increase the SOC of a vehicle. While this sounds undesirable at first glance, we observe that such recharging cycles are only included if they lead to a feasible solution with overall minimum travel time. As we discuss in Section 5.4, such cycles did occur rarely in our experiments and only in case of extremely small battery capacities and very few charging facilities.

Notation. 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 $\beta(a, t, b) = -\infty$ if it is infeasible to cross the arc a with travel time t and initial SOC b. Since the depot can be visited multiple times in a single walk, we 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 $b \in [0, Q]$.

Example. 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), and $(3, 1)$. A travel time of 2 is chosen on all other arc traversal not adjacent to the artificial depot.

2 Integer programming formulation

As discussed above, eARP instances may contain *positive SOC cycles* 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. 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., Belenguer et al. [10]. 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. Bartolini et al. [6] and Section 3.1 for further details. Hence, such a transformation would result in creating a multi-graph with a potentially exponential number of parallel edges, which

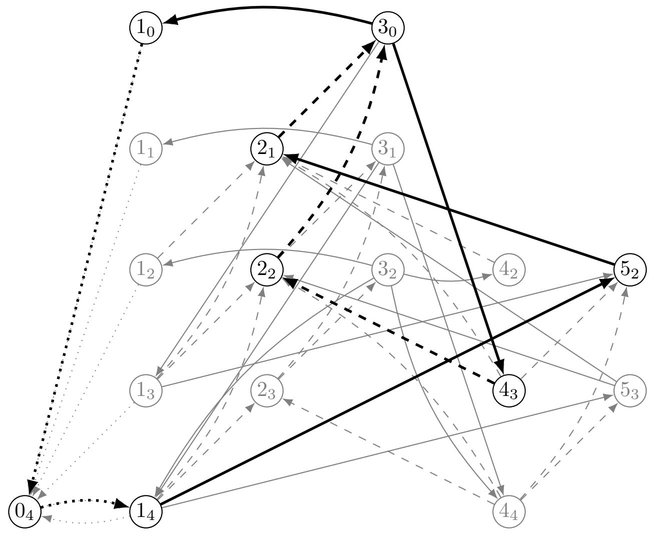


Figure 2: Energy-indexed graph corresponding to the instance given in Figure 1a. The solution given in Figure 1b is indicated by bold arcs. Artificial arcs adjacent to 0_4 are shown by dotted lines, optional arcs by dashed lines, and arcs corresponding to required arcs by solid lines. Nodes without ingoing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

might not be tractable from a computational perspective. We overcome these difficulties by proposing a transformation of the eARP to an equivalent problem defined on an *energy-indexed graph*.

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

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 of arc $(u, v) \in A$. Above definition of $\mathcal{G}$ is based on introducing one copy u_p of each node u at each possible SOC $p \in B$. Our implementation is, however, based on a (usually) much smaller graph that only contains nodes that are reachable via an energy-feasible walk from the artificial depot 0 (see Section 4.1 for details). It is clear that $\mathcal{G}$ is notably larger than 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 (whose proof is given in Appendix A), 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.*

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 $\mathcal{A}_{uv}$ is traversed for each required arc $(u, v) \in A_{\mathrm{R}}$. Thus, the eARP can be seen as a particular case of the GDRPP defined on $\mathcal{G}$, 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

The ILP formulation exploits the structure of the energy-indexed graph $\mathcal{G}$. It considers the following sets of decision variables: (i) integer variables $X_a, \forall a = (u_p, v_q) \in \mathcal{A}$, 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 (ii) binary variables $y^p_{uv}, \forall (u, v) \in A_{\mathrm{R}}$, $p \in B$, that indicate if required arc (u, v) is served with SOC level p at node u. We use compact notation $X(\mathcal{A}') = \sum_{a \in \mathcal{A}'} X_a$ for any set of arcs $\mathcal{A}' \subseteq \mathcal{A}$ as well as notations $\delta^+(S) = \{(u_p, v_q) \in \mathcal{A} \mid u_p \in S\}$ and $\delta^-(S) = \{(u_p, v_q) \in \mathcal{A} \mid v_q \in S\}$ for $S \subset \mathcal{V}$. Thereby, we omit set braces when S contains only one node.

$$
\begin{aligned}
\min \quad & \sum_{a \in \mathcal{A}} t_a X_a && && \text{(1a)}\\
\text{s.t.} \quad & X(\delta^+(0_Q)) \leq m && && \text{(1b)}\\
& X(\mathcal{A}_{uv}) \geq 1 && \forall (u, v) \in A_{\mathrm{R}} && \text{(1c)}\\
& X(\delta^-(u_p)) = X(\delta^+(u_p)) && \forall u_p \in \mathcal{V} && \text{(1d)}\\
& \sum_{p \in B} y^p_{uv} = 1 && \forall (u, v) \in A_{\mathrm{R}} && \text{(1e)}\\
& X(\{(u_p, v_q) \in \mathcal{A}\}) \geq y^p_{uv} && \forall (u, v) \in A_{\mathrm{R}}, p \in B && \text{(1f)}\\
& X(\delta^-(S)) \geq \sum_{u_p \in S} y^p_{uv} && \forall (u, v) \in A_{\mathrm{R}}, S \subseteq \mathcal{V} \setminus \{0_Q\} && \text{(1g)}\\
& X_a \in \mathbb{Z}_+ && \forall a \in \mathcal{A} && \text{(1h)}\\
& y^p_{uv} \in \{0, 1\} && \forall (u, v) \in A_{\mathrm{R}}, p \in B. && \text{(1i)}
\end{aligned}
$$

Inequality (1b) ensures that no more than m walks leave the artificial depot 0_Q, while constraints (1c) guarantee that at least one arc from each set of arcs corresponding to a single required arc in G is traversed. Equations (1d) are flow conservation constraints. Assignment constraints (1e) ensure that each required arc is served exactly once. Linking inequalities (1f) 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 (1g) together with (1d)-(1f). Whenever arc $(u, v) \in A_{\mathrm{R}}$ is served with some initial SOC p, connectivity constraints (1g) 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. Constraints (1g) are valid since equations (1e) imply that the right-hand side is at most one. Note that the feasible domain of formulation (1) 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 (1a) minimizes the overall travel times. We also observe that constraints (1c) can be obtained as a linear combination of (1e) and (1f). We keep them in the model, however, as in our solution framework not all constraints (1f) are present in the initial formulation (see Section 4).

Some of the coefficients in the connectivity constraints (1g) 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 (2) in which the coefficients of the arcs from $\delta^-(S)$ that must not be used are down-lifted to zero.

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

A particular special case of connectivity constraints (1g) 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 (1g) can be re-written as

$$
X(\delta^-(\mathcal{V}(W))) \geq 1 \qquad \forall W \subseteq V \setminus \{0\} : W \cap \{u \in V : (u, v) \in A_{\mathrm{R}}\} \neq \emptyset. \tag{3}
$$

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

We observe, that formulation (1) can be easily extended to a variant of eARP including (undirected) demand edges. For each required edge $\{u, v\}$, arcs in both directions need to be considered in a corresponding energy-indexed graph for each relevant SOC level $p \in B$ together with individual serving variables y^b_{uv} and y^b_{vu}. Finally, X and y variables for both serving directions need to be included on the left-hand sides of constraints (1c) and (1e), respectively.

3 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 4), or as stand-alone approaches for dealing with large-scale instances. All construction and improvement heuristics proposed in Section 3.2 operate on the solution representation presented in Section 3.1.

3.1 Solution representation and decoding

We represent each feasible solution by the sequence of required arcs served by each vehicle, similar to Vidal [40]. That way, a sequence $\mathcal{R} = (a^1, a^2, \dots, a^r)$, $a^j \in A_{\text{R}}$, $j \in \{1, 2, \dots, r\}$, 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. The proof of Theorem 1 (see Appendix A) shows how to identify whether a given sequence of required arcs leads to a feasible solution or not.

Theorem 1. *For a given sequence $\mathcal{R} = (a^1, a^2, \dots, a^r)$ 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.*

We next develop a labeling algorithm that calculates a walk with an associated sequence of travel times for a given sequence $\mathcal{R} = (a^1, a^2, \dots, a^r)$, $a^j = (u^j, v^j) \in A_{\text{R}}$, $j = 1, \dots, r$, of required arcs 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 [26]. In each iteration the algorithm computes a set of labels $\mathcal{L}[v^j]$ corresponding to non-dominated walks from the depot 0 to v^j while taking the label set $\mathcal{L}[v^{j-1}]$ as input, see Algorithm 1. Each label $(t, b, \mathcal{W}, \mathcal{T})$ is represented by its associated walk $\mathcal{W} = (a^1, a^2, \dots, a^k)$, $a^l = (w^{l-1}, w^l)$ with $w^0 = 0$, travel time sequence $\mathcal{T} = (t^1, t^2, \dots, t^k)$, $t^l \in T(a^l)$ for $l \in \{1, 2, \dots, k\}$, total travel time $t = \sum_{l=1}^{k} t^l$, and final SOC $b \in B$ at node w^k. A label $(t, b, \mathcal{W}, \mathcal{T})$ is dominated by another label $(t', b', \mathcal{W}', \mathcal{T}')$ if and only if both walks end at the same node w^k, serve the same partial sequence of required arcs, $t \geq t'$, and $b \leq b'$. The algorithm is initialized with label $\mathcal{L}[v^0] = \{(0, Q, \emptyset, \emptyset)\}$ corresponding to the empty walk starting and ending at the depot $0 =: v^0$. In its final iteration, the algorithm extends all non-dominated walks from $L[v^r]$ to the depot $0 =: u^{r+1} = v^{r+1}$, and returns the label with smallest travel time, i.e., $(t^*, b^*, \mathcal{W}^*, \mathcal{T}^*) \in \operatorname{argmin}_{(t,b,\mathcal{W},\mathcal{T}) \in \mathcal{L}[v^{r+1}]} t$, or states infeasibility in case $\mathcal{L}[v^{r+1}] = \emptyset$. Algorithm 1 uses operator $\circ$ to append an arc to a walk and a travel time to a sequence of travel times. Operator $\oplus$ is used for possibly inserting a new label: If the 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. Parameter $t_{\max}$ indicates the maximum travel time and is used to restrict the label extension. Observe that an upper bound (we use the travel time of the currently best known solution) can be tightened to $t_{\max} - \sum_{k=j}^{r} \min T(a^k)$ in iteration j by assuming that each required arc of sequence $\mathcal{R}$ will be traversed with the minimum possible travel time.

3.2 Construction and improvement heuristics

We propose three different construction heuristics all of which represent solutions as sequences of required arcs and use the algorithm described in Section 3.1 for their decoding and evaluation.

Algorithm 1: ExtendWalks($\mathcal{L}[v^{j-1}], (u^j, v^j), t_{\max}$)

```
Input: labels $\mathcal{L}[v^{j-1}]$ corresponding to non-dominated walks from 0 to $v^{j-1}$, subsequent
       required arc $(u^j, v^j) \in A_R$, travel time limit $t_{\max}$
Output: labels $\mathcal{L}[v^j]$ corresponding to non-dominated walks from 0 to $v^j$
1  $\mathcal{L} = \mathcal{L}[v^{j-1}]$
2  foreach unprocessed label $(t, b, \mathcal{W}, \mathcal{T}) \in \mathcal{L}$ do          // compute labels to $u^j$
3      foreach $a \in \delta^+(u)$ and $t' \in T(a)$ do     // node $u$ denotes the final node in walk $\mathcal{W}$
4          if $t + t' \leq t_{\max}$ and $\beta(a, t', b) \in B$ then $\mathcal{L} = \mathcal{L} \oplus (t + t', \beta(a, t', b), \mathcal{W} \circ a, \mathcal{T} \circ t')$
5  if $(u^j, v^j) = (0, 0)$ then                                  // return back to the depot
6      $\mathcal{L}[v^j] = \{(t, b, \mathcal{W}, \mathcal{T}) \in \mathcal{L} \mid \mathcal{W} = (a^1, a^2, \dots, (w, u^j))\}$
7  else                                    // extend all labels at $u^j$ along $a^j = (u^j, v^j)$
8      $\mathcal{L}[u^j] = \{(t, b, \mathcal{W}, \mathcal{T}) \in \mathcal{L} \mid \mathcal{W} = (a^1, a^2, \dots, (w, u^j))\}$
9      $t_{\max} = t_{\max} + \min_{t \in T(a^j)} t$
10     foreach $(t, b, \mathcal{W}, \mathcal{T}) \in \mathcal{L}[u^j]$ and $t' \in T(a^j)$ do
11         if $t + t' \leq t_{\max}$ and $\beta(a^j, t', b) \in B$ then
12             $\mathcal{L}[v^j] = \mathcal{L}[v^j] \oplus (t + t', \beta(a^j, t', b), \mathcal{W} \circ a^j, \mathcal{T} \circ t')$
13 return $\mathcal{L}[v^j]$
```

Heuristic *SM*. This heuristic is inspired by the savings heuristic for the capacitated VRP [13] and the Merge heuristic for the capacitated ARP [21]. 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. 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 that 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 further merge would increase the travel time. Merges that increase the objective value may be performed if 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 [8]. It first computes a giant route of minimal total travel time serving all required arcs but ignoring the battery capacity. We observe that an optimal solution to a relaxation of the eARP that neglects the battery constraints can be computed by solving a DRPP instance 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 with 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 [5]. This approach is feasible since the DRPP using only the lowest travel times can be solved very fast even for the most difficult eARP instances considered in our study. An optimal DRPP solution also provides a valid lower bound on the optimal value of the associated eARP instance. 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. [24]. For each pair $k, l \in \{1, 2, \dots, |A_R|\}, k < l$, the optimal walk for the subsequence $(a^k, a^{k+1}, \dots, a^l)$ is computed by the labeling algorithm described in Section 3.1. The combination of the best set of sequences for $(a^1, a^2, \dots, a^{k-1})$ together with the optimal walk for $(a^k, a^{k+1}, \dots, a^l)$ 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. The merge heuristic *SM* is finally applied to its result in case the number of derived routes exceeds the given number of vehicles m. Since each split sequence of required arcs is optimally decoded, Corollary 1 follows from arguments analogous to those used in Lacomme et al. [24] and Ulusoy [39] 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). A best improvement strategy is used and the local search algorithm stops if there is no move that improves the objective function value.

Acceleration techniques. The decoding of sequences of served arcs may induce long running times in case many Pareto-optimal walks exist. Three techniques addressing this issue are considered in our 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.

As detailed in Section 3.1, 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 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 [36], Ruthmair and Raidl [37]. This data structure allows to efficiently insert sets for newly computed sequences and find the longest already computed prefix for a given sequence. Thus, the labeling algorithm described in Section 3.1 can skip the required arcs contained in the found prefix and hot-starts from the set of Pareto-optimal walks stored for this prefix.

4 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.

4.1 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 energy-indexed nodes or arcs that cannot be obtained in an eARP solution. Instead, we incrementally create $\mathcal{G}$ by only considering node-SOC pairs reachable in an energy-feasible walk starting at the depot. 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. At the end the energy-indexed graph is reduced by removing nodes and arcs that cannot appear in an optimal solution. The following three rules are applied repeatedly until no further reductions can be made: (i) Nodes without outgoing arcs cannot be part of a closed walk and are thus removed together with all incoming arcs. (ii) Cycles containing only non-required 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 $(u, v), (v, u) \in A \setminus A_{\mathrm{R}}$. 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 $q' \geq q$. (iii) Nodes without incoming arcs are removed together with all outgoing arcs.

4.2 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 (1d) 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. We first separate constraints (3) rewritten with x variables in original graph G. If no such constraints are violated in the current iteration, constraints (1g) or their stronger variants (2) are separated. The latter two are only added if the *cut violation* ratio between left-hand and right-hand side values 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 the energy-indexed graph cuts for fractional solutions only at the root node of the branch-and-cut tree with a limit of at most 100 inequalities in each iteration. Next, we describe how we identify violated inequalities (1f), (1g), (2), and (3) for fractional solutions. Let $\bar{x}$, $\bar{X}$, and $\bar{y}$, denote the variable values of the current LP relaxation.

Separation of constraints (1g) **and** (2)**.** Using equations (1e) we can rewrite constraints (1g) as

$$X(\delta^-(S)) + \sum_{u_p \notin S} y^p_{uv} \geq 1 \qquad \forall (u, v) \in A_{\mathrm{R}}, S \subseteq \mathcal{V} \setminus \{0_Q\}. \tag{4}$$

We find a minimum cut for a given required arc $(u, v) \in A_{\mathrm{R}}$ by introducing an artificial target node t and connecting 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 induces a violated inequality (4). To separate the lifted variant (2) for a given 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.

Since the number of linking constraints (1f) connecting y and X variables can be rather high, we separate them dynamically only for those y variables contained in some violated cut (1g) or (2).

Separation of constraints (3)**.** Violated connectivity constraints (3) can be identified in the original graph G. For each node $u \in V$ that 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 (3).

To obtain a minimum cut of smallest cardinality (leading to a sparser inequality) we add a small value, i.e., 10^{-5}, to all arc capacities, cf. Fischetti et al. [17]. To further break ties we choose a cut that has a smallest set S. For integer solutions inequalities (1f), (1g), (2), and (3) are separated efficiently using breadth-first search. For (1g) and (2), we only add a single violated inequality (plus linking constraints (1f) if applicable) to cut-off infeasible solutions.

5 Experimental results

In this section we describe and motivate our test instances, and evaluate the performance of our algorithms. Finally, we perform experiments with varying battery sizes and network properties to derive managerial insights and recommendations. Each experiment has been performed on a single core of an Intel Xeon E5-2670v2 machine with 2.5 GHz with a memory limit of 8 GB.

Due to the lack of available data for other arc routing applications with electric devices such as, e.g., inspection robots, we generate our benchmark instances focusing on applications in transportation and logistics.

5.1 Instances

For constructing realistic 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. The street networks are obtained by using the arc routing instance generator OARBench [28]. 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 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 we keep the longest one. Nodes that 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 that are within a Euclidean distance of 500, 300, or 100 meters, resulting in three graphs with different sizes for each city. The 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 10% and 50% of all arcs, respectively. Instance names consist of the city's first letter and $|V|$-$|A|$-$|A_\mathrm{R}|$, e.g., A-31-105-8.

We compute the vehicle's energy consumption as in Asamer et al. [4], i.e., by reformulation we obtain for travel time t and speed v, $e(t, v) = t\left(\gamma_1 v + \gamma_2 v^3 + \gamma_3\right)$, where $\gamma_1, \gamma_2, \gamma_3$ are vehicle-specific constants: Based on the specifications of an Eforce EF18 electric truck (`www.eforce.ch/products/ef18`, 2018-11-20), 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 $\gamma_1 = \frac{g \cdot C_r \cdot m}{\eta} = \frac{9.81 \cdot 0.008 \cdot 15000}{0.97} \approx 1213.61$, $\gamma_2 = \frac{C_w \cdot A_\mathrm{F} \cdot \rho}{2 \cdot \eta} = \frac{0.8 \cdot 9.69 \cdot 1.165}{2 \cdot 0.97} \approx 4.66$, and $\gamma_3 = P_0 = 3000$, where g is gravity, C_r rolling resistance, m weight, η engine efficiency, C_w drag coefficient, A_F frontal surface area, ρ air density (200m above sea level, temperature 20°C), and P_0 auxiliary power. For a fixed speed the energy consumption is linear in t, i.e., $e(t, v) = \hat{e}(v)t$ with $\hat{e}(v) := \gamma_1 v + \gamma_2 v^3 + \gamma_3$. We can derive the speed to drive some distance d that leads to the minimal energy consumption by replacing t with d/v, i.e., $v^* = \sqrt[3]{\gamma_3/(2\gamma_2)} \approx 24.7$ km/h. With this speed the truck needs about 1870 kWs $\approx$ 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 with up to 630 kWh. Since the battery is one of the major factors in the truck acquisition costs and there are environmental concerns related to its production and recycling, we want to keep it as small as possible and it turned out that already small sizes are sufficient in our cases, see Section 5.4.

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. Montoya et al. [31]. 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 $\hat{c}(t) = \min\{r_1 t,\ 0.85Q + r_2(t - t_1),\ 0.95Q + r_3(t - t_2),\ Q\}$, 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 Baum et al. [7] 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 dynamic 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

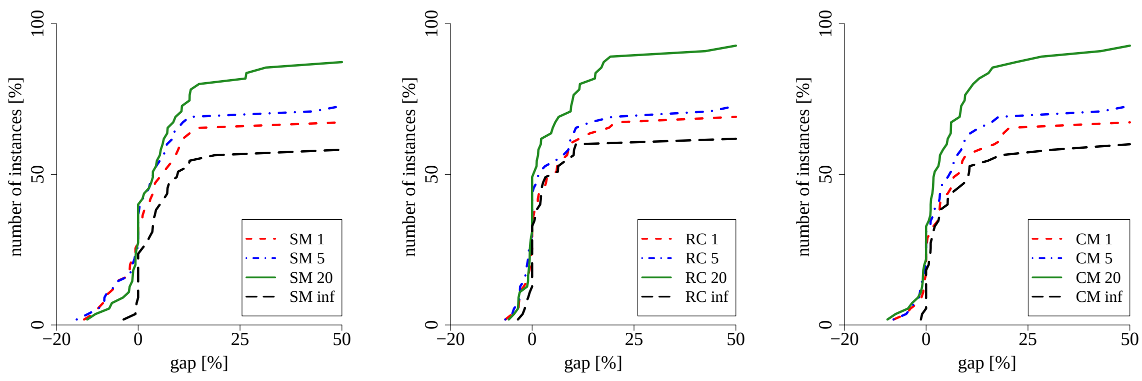


Figure 3: Gaps of heuristic solutions compared to best branch-and-cut solutions.

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 dynamic charging technology allows to re-charge with 22 kW at any of these speeds [11]. Finally, the SOC function $\beta(a,t,b)$ on arc $a \in A$, for travel time t, and initial SOC b is defined as

$$\beta(a,t,b) = \begin{cases} c(t,b) & \text{if } a \text{ is a charging arc and } r_1(\frac{d_a}{t}) > 0, \\ b + r_1(\frac{d_a}{t})t & \text{if } r_1(\frac{d_a}{t}) \leq 0 \text{ and } b + r_1(\frac{d_a}{t})t \geq 0, \\ -\infty & \text{otherwise,} \end{cases} \tag{5}$$

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.

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

In this section we report the performance of the heuristics given in Section 3 and the exact solution framework described in Section 4 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 (see the paragraph on acceleration techniques in Section 3). 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 3 depicts cumulative gaps of the heuristics for values of $K \in \{1, 5, 20, \infty\}$, while Figure 4 shows their cumulative runtimes in seconds. Heuristic solution quality 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) 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 heuristic RC with $K = 20$ without local search). A closer look into the results, see Appendix B, 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 ($K \leq 5$), no solution could be found in some cases. However, Figure 3 clearly indicates that higher values of K do not necessarily imply a better solution quality. This can be explained by the runtime increase for growing values of K due to the additional effort in sequence decoding (see Section 3.1). 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 B. 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

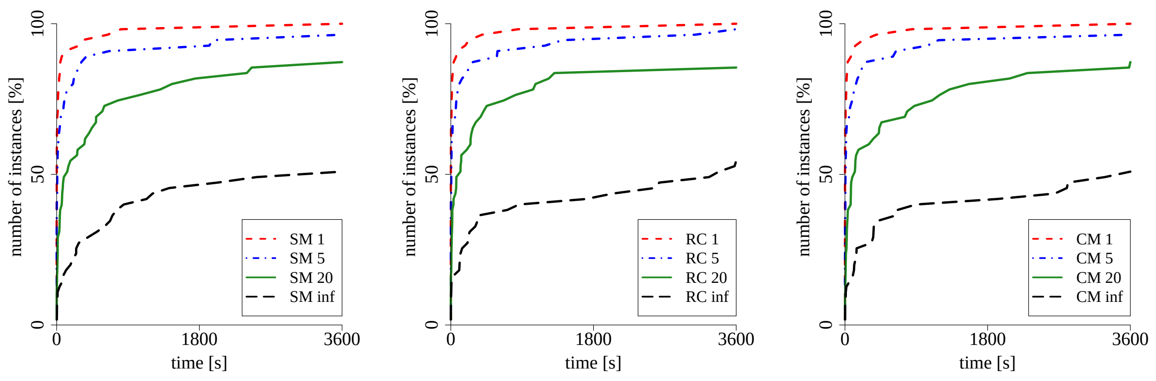


Figure 4: Solution times of heuristics.

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 difficult instances with a high number of required arcs (with few 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.

The usage of the archive produces significant savings of computing time 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 archive seem to provide a good compromise between solution quality and runtime. Details are discussed in Appendix B.

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

- Variant BC uses the first visit variables y, adds (1e) and (1i) statically to the formulation and separates constraints (1f) and lifted inequalities (2) dynamically for integer solutions only.
- Variant BC+ is an extension of BC by separating fractional points as well and adding inequalities (2) to strengthen the dual bound of the LP relaxation (only) at the root node.

Both variants use the base formulation (1a)-(1d) together with (1h), and separate connectivity cuts (3) 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 and handed over to CPLEX as a first incumbent solution (if available).

Figure 5 compares cumulative gaps and computing times in seconds of variants BC and BC+. More detailed results are given in Table 1 that 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, and the total number of branching nodes. For each of the three heuristics (with default setting $K = 20$, enabled archive, and local search), we report the relative gap to the best upper bound UB. The runtime of each test run is limited to 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 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

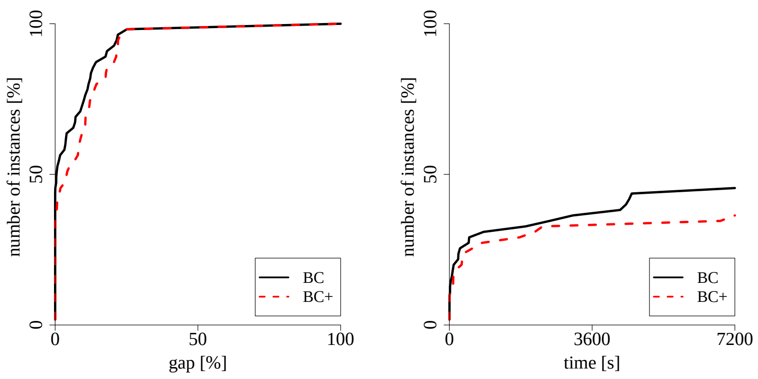


Figure 5: Final optimality gaps and computing times for branch-and-cut approaches.

nodes only. In general, the base energy-indexed graph formulations with connectivity cuts (3) but without inequalities (2) are already quite strong, as also observed in Gouveia et al. [22] for layered graph formulations for other problems. The 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 quite good but the primal bounds are mostly far from the optimum. 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 heuristics applied in branch-and-bound nodes 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 the number of branching nodes: While both settings seem to achieve similar quality of solutions, 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 tends to achieve better lower bounds and smaller branching trees.

Finally, we point out that there are no infeasible instances in this set but with different parameters 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 3.1. 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.

5.3 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 ILP 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). In the following study, we consider three different discretization levels, $D \in \{500, 1000, 2000\}$ in kWs, on 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:

- Fixed sequence of traversed arcs, $R(D, D')$: We fix the walks $\mathcal{W}_1, \ldots, \mathcal{W}_m$, obtained with D and re-calculate an optimal sequence of travel times with respect to the discretization level D'. To this end, for each walk $\mathcal{W}_i = (a^1, a^2, \ldots, a^k)$, $a^j = (u^{j-1}, u^j) \in A$, $1 \leq j \leq k$, the dynamic programming recursion $\tau(u^j, b) = \min_{t \in T(a^j), b' \in B:\ \beta(a^j, t, b') = b} \{\tau(u^{j-1}, b') + t\}, \quad \forall j = 1, \ldots, k, \forall b \in B$, is used to calculate the minimum travel time $\tau(u^j, b)$ for arriving at node u^j with SOC b. We initialize $\tau(u^0, Q) = 0$ and $\tau(u^0, b) = \infty$ for all $b \neq Q$. Observe that the travel time obtained with D is an upper bound for the travel time based on the refined discretization level D'.

- Fixed 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 the decoding algorithm (with no limit on the number of labels) based on discretization level D' 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 $OPT_g(D) = \frac{OPT(D) - OPT(500)}{OPT(500)}$, $S_g(D, D') = \frac{S(D, D') - OPT(500)}{OPT(500)}$, $R_g(D, D') = \frac{R(D, D') - OPT(500)}{OPT(500)}$. 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), the decoding algorithm (which is applied with no upper limit on the number of labels) needed for the calculation of $S(D, D')$, see Section 3.1, can take a significant amount of time, cf. Section 5.2, 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 over-estimation 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.

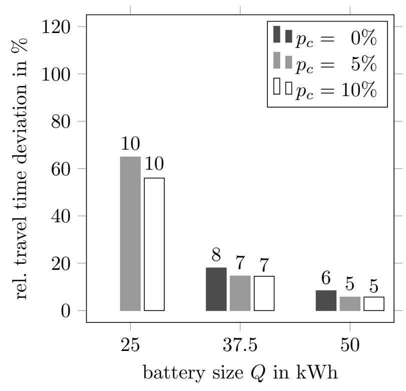
It is interesting to see that re-evaluating the travel times using method S rarely leads to better solutions than method R indicating that the walks usually do not change after applying method S. Only for instances A-58-168-15 and A-84-220-20, battery size $Q = 25$, and discretization level $D = 2000$, we can observe a slight improvement when using method S.

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

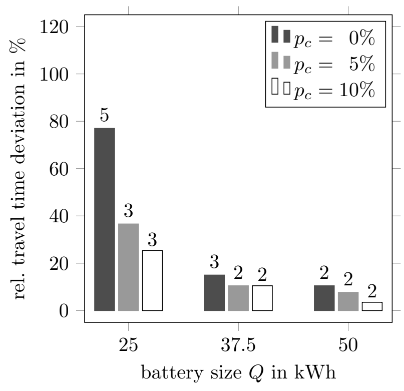
5.4 What are the costs of reduced battery capacities?

The battery is a major part of the vehicle acquisition costs and there are environmental concerns in its production and recycling. Thus, reducing its capacity to a reasonable size customized to a particular application is highly desired. Smaller batteries may be facilitated by dynamic charging along the arcs, which eliminates waiting times at stations. Hence, one may ask whether the increase of the total travel time arising from equipping vehicles with batteries having comparably small capacity can be sufficiently compensated by existing dynamic charging infrastructure. Related to that, one may wonder whether small batteries or sparse charging infrastructure lead to solutions in which additional cycles are driven only for re-charging the battery, cf. Section 1.2. To obtain insights into these aspects, we analyze the obtained optimal solutions for different battery sizes and relative amounts of dynamic charging arcs.

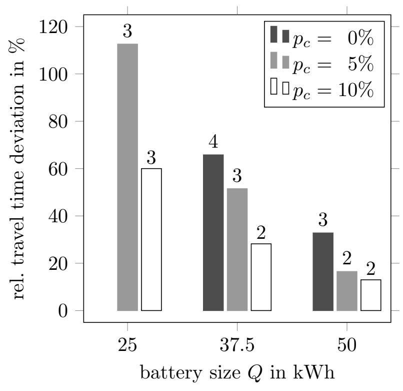
Figure 6 compares optimal solutions for three exemplary instances, different battery sizes $Q \in \{25, 37.5, 50\}$, and different charging infrastructure, obtained with the default discretization level of 1000 kWs and re-evaluated with method R, see Section 5.3, based on a more fine-grained discretization of 1 kWs. We report the relative deviations $(R(1000, 1) - OPT_D)/OPT_D$ (in percent) from the optimal



(a) instance A-31-105-47



(b) instance P-61-205-20



(c) instance V-60-218-20

Figure 6: Solution comparison for different battery sizes and numbers of charging facilities.

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 dynamic 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 5.1. The values above the bars in Figure 6 present the number of vehicles used in the obtained solution.

We first observe that only very few of the optimal solutions obtained for these settings contain cycles without required arcs that are driven for re-charging the battery. In fact, only in 2 (instance V) cases (of 162) with the smallest battery capacity 25 kWh such re-charging cycles appeared in an optimal solution.

Next, we study the effect of the battery capacity on the increase of travel times, compared to a fleet with combustion engines. As shown in Figure 6, 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%. These numbers significantly drop down by increasing the battery capacity, improving the infrastructure, or both. In the most optimistic scenario we tested ($Q = 50$ kWh, $p_c = 10\%$), the increase of the total travel time is almost negligible, and ranges from 3.5% (instance P) to 13% (instance V).

We also 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 routes. We do not limit the number of vehicles in our tests to see how optimal solutions are structured, which means that the fleet size is implicitly optimized in our models. So, if a larger battery or a better infrastructure allows to reduce the fleet size, 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).

6 Conclusions

In this article we proposed 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. A 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. We propose to encode solutions as sequences of required arcs, study the computational complexity of verifying their feasibility, and introduce a labeling algorithm for solution decoding. 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.

Our results also highlight the trade-off between investment costs for the charging infrastructure and the potential increase of travel times due to limited battery range. These findings naturally lead to other interesting optimization problems related to strategic decisions such as network design of charging infrastructure or simultaneous optimization of battery size and charging infrastructure. The further study of such optimization problems and the development of corresponding decision-support tools can help with the adoption of dynamic charging technology in passenger and cargo transportation.

Acknowledgements

E. Fernandez was partially supported by the Spanish Ministry of Economy and Competitiveness through MINECO/FEDER grants MTM2015-63779-R and PID2019-105824GB-I00. M. Leitner, I. Ljubić, and M. Ruthmair were partially funded by the Vienna Science and Technology Fund through project ICT15-014.

References

[1] A. Ahmad, M. S. Alam, and R. Chabaan. A comprehensive review of wireless charging technologies for electric vehicles. *IEEE Transactions on Transportation Electrification*, 4(1):38–63, 2018.

[2] J. Allan and J. Beaudry. Robotic systems applied to power substations - a state-of-the-art survey. In *Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry*, pages 1–6, 2014.

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

[4] 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.

[5] 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.

[6] 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.

[7] M. Baum, J. Dibbelt, A. Gemsa, D. Wagner, and T. Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. *Transportation Science*, 53(6):1627–1655, 2019.

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

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

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

[11] 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.

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

[13] 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.

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

[15] 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.

[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] A. Froger, J. E. Mendoza, O. Jabali, and G. Laporte. Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Computers & Operations Research*, 104:256–294, 2019.

[20] 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.

[21] 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.

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

[23] IPT Technology. IPT Technology, 2020. URL `https://ipt-technology.com/`. accessed on 2020-06-28.

[24] 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.

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

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

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

[28] 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.

[29] 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.

[30] O. Menéndez, F. A. Auat Cheein, M. Perez, and S. Kouro. Robotics in power systems: Enabling a more reliable and safe grid. *IEEE Industrial Electronics Magazine*, 11(2):22–34, 2017.

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

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

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

[41] B. Wang, R. Guo, B. Li, L. Han, Y. Sun, and M. Wang. Smartguard: An autonomous robotic system for inspecting substation equipment. *Journal of Field Robotics*, 29(1):123–137, 2012.

[42] Wiferion. Wiferion, 2020. URL `https://ipt-technology.com/`. accessed on 2020-06-28.

A Proofs

Proof (Lemma 1). Let $\mathcal{W} = (a^1, a^2, \ldots, a^k)$, $a^j = (u^{j-1}, u^j) \in A$, $1 \leq j \leq k$, $u^0 = u^k = 0$, be a battery feasible walk in G and $\mathcal{T} = (t^1, t^2, \ldots, t^k)$, $t^j \in T(a^j)$, $1 \leq j \leq k$, be the associated travel times. For each j, $1 \leq j < k$, 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. □

Proof (Theorem 1). 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 $\bar{t}(a) \in T(a)$ that 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 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. □

B Heuristic results

In Table 3 we compare the algorithmic performance of our heuristic methods for the eARP from Section 3. The three construction heuristics SM, RC, and CM, including local search are applied to the city instances with a time limit of one hour and a memory limit of 8 GB. The quality of some heuristic solution

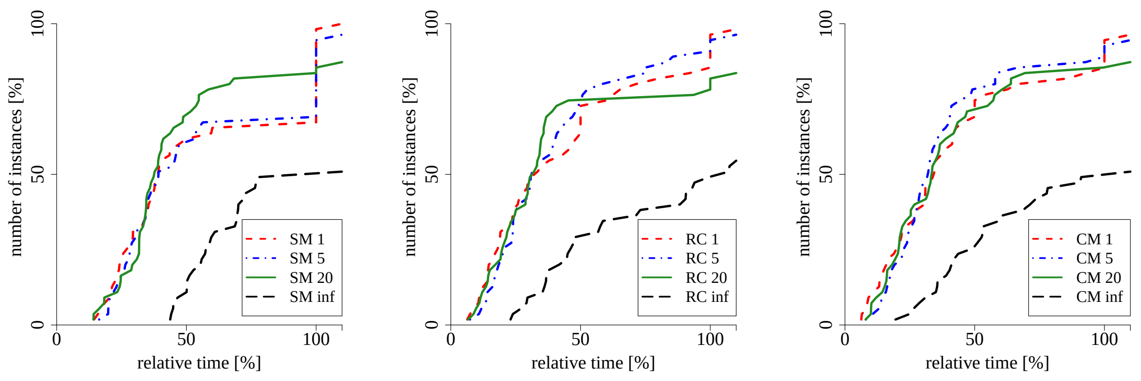


Figure 7: Solution times of heuristics with solution archive relative to variants without archive.

is measured as the relative travel time deviation to the best solution found by the BC approaches in Section 5.2. 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.

Figure 7 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 the sequences are iteratively extended at the end allowing to re-use the labels stored in the archive from previous iterations.

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**

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

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

			relative primal bound in %												runtime in seconds												archive speedup $t_{\mathrm{noA}}/t_{\mathrm{A}}$											
			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	-	-	-