Location of charging stations in electric car sharing systems

Georg Brandstätter[1], Markus Leitner[1], and Ivana Ljubić[2]

[1]Department of Statistics and Operations Research, University of Vienna, Vienna, Austria,
georg.brandstaetter@univie.ac.at, markus.leitner@univie.ac.at
[2]ESSEC Business School of Paris, Cergy Pontoise, France, ivana.ljubic@essec.edu

October 23, 2018
previous version: November 5, 2016

1 Introduction

Urbanization is an ongoing process that makes more and more people move from rural areas to nearby cities and suburbs. This results in an increased demand for mobility in these areas, much of which is currently satisfied by privately owned cars powered by internal combustion engines. At the same time, people are becoming more concerned about sustainability and the effects of pollution on the environment, of which the aforementioned vehicles are a major cause. While public transportation can help (and has helped) to mitigate some of the negative effects resulting from such private car ownership, their inflexibility inherently limits their usefulness (Jorge and Correia 2013). Thus, cities need to offer their residents new solutions for individual transportation that are environmentally friendly, flexible and affordable. *Car sharing* systems using *electric vehicles* are one example of such a solution that has recently attracted increasing attention.

Car sharing systems operate on the concept of a fleet of vehicles that can be rented by customers for short periods of time. This allows fewer vehicles to satisfy the mobility demands of more people, thereby reducing the amount of public space required (Shaheen, Sperling, and Wagner 1998). Using electric vehicles allows these systems to operate in an environmentally friendly manner, as they are very efficient in urban settings and do not produce tailpipe emissions, which helps eliminate (if the electricity comes from clean sources such as hydro, wind or solar power) or at least mitigate air pollution by moving it to less densely populated areas (Pelletier, Jabali, and Laporte 2014).

However, despite recent technological advancements, the range and recharge speed of electric vehicles, especially economically viable ones, is still fairly limited (Pelletier, Jabali, and Laporte 2016). A "smart electric drive", for instance, has a range of only around 145 km and requires one whole hour to be fully recharged, even with 22 kW fast-charging infrastructure (see, e.g., Daimler AG (2012)). Similarly, the "Mitsubishi iMiEV" and the "Nissan LEAF", both with a range of around 160 km, require about 30 minutes to recharge their batteries by approximately 80% Mitsubishi (2017), Nissan (2014). However, since the charging characteristics of their batteries are highly non-linear, the remaining 20% are charged more slowly.

Thus, in order to remain operational, these vehicles must frequently be recharged for relatively long periods of time when they are not in use, which happens at *charging stations*. Such charging stations, however, are very expensive to build, especially if they allow faster recharge speeds than common household power outlets. A study conducted by Gawlik et al. (2013) with data from Vienna has, for instance, suggested that the cost of a single charging station can range between 56500 and 126500 Euro, largely depending on the cost of the station terminal and the length of the power supply cable for which trenches must be dug.

To make the best use of available funds, the locations and capacities of those stations must therefore be chosen carefully. One obvious criterion for determining the viability of a potential charging station location is its proximity to customer demand. Likewise, the amount of nearby demand determines which size (capacity) would be best for each station. Thus, an important prerequisite for any optimization approach aiming to identify optimal locations and capacities of charging stations is the availability of an appropriate forecast of the expected demand throughout the operational area. Important characteristics of such a demand forecast (or a method generating such a forecast) in the context of electric car sharing are the consideration of (i) the demand variance over the course of the planning period and (ii) the expected battery consumptions to facilitate the planning of necessary recharging breaks of vehicles. In addition, it should account for the movement of cars between different stations / parts of the operational area during the planning period. One way of incorporating these characteristics is using a *forecast of estimated trips* as demand model. An additional advantage of this choice (that is taken in this article) is that it allows one to easily associate estimated profits to the individual trips of such a forecast, thus allowing one to optimize the expected profit in addition to other important strategic optimization objectives, such as cost minimization while satisfying the complete (or a certain fraction of the) estimated demand.

In this work, we define an optimization problem for finding optimal locations and sizes for charging stations within an electric car sharing network's operational area while using such a trip forecast. We focus on maximizing the expected profit or number of accepted trips while explicitly considering the charge state of the individual vehicles, i.e., ensuring that their batteries are never depleted when fulfilling the demand.

While we are mainly concerned with the strategic decisions related to the placement and sizing of charging stations, the models and algorithms we present also optimize decisions at lower levels (e.g., fleet sizing, trip selection and their assignment to vehicles) to improve the quality of the higher-level decisions. Such integrated optimization approaches have already been successfully used for other problems like the production-routing problem (see, e.g., Ruokokoski et al. (2010), Absi et al. (2018)). In this study, we show that significant improvements of the expected profit can be achieved when lower-level decisions are taken into account, even under the assumption of uncertain customer demand.

Outline. We summarize previous and related work in the remainder of this section. In Section 2, we formally define the optimization problem we want to solve. We prove in Section 3 that the problem is NP-hard in general and in two relaxed variants, but also show that an even more relaxed problem variant can be solved in polynomial time. Section 4 describes the integer linear programming (ILP) formulations that we used to model and solve the problem. Two heuristic methods are introduced in Section 5, where we also describe our separation procedures for dynamically finding constraints for the ILP models introduced in Section 4 during branch-and-cut. In Section 6, we compare the performance and solution characteristics of our algorithms on grid-graph-based instances, as well as ones based on real-world data from the city of Vienna. Section 7 describes a sequential algorithm that considers decisions at the various levels separately and compares it to the previously-described integrated models where these decisions are optimized together. We also conduct a cross-validation study on a second set of real-world instances from Vienna to analyze the behavior of our algorithms in cases where demand is uncertain, and describe an algorithm for computing a *consensus* solution from the optimal solutions for the individual deterministic instances. In Section 8, we then conduct a simulation study to analyze the behavior of our solutions in a car-sharing system where trip requests are fulfilled on a first-come-first-served basis and customers are free to choose any available car and charger for their trip. Finally, Section 9 contains concluding remarks and an outline for future work.

State of the art. The domains of both car sharing and electric vehicles have recently attracted increased interest from the scientific community. The literature on strategic optimization problems arising in the context of their combined use is still fairly sparse, however. In the following we summarize these works and also give a brief overview on further, related problems. A more detailed classification and overview of (recent) related scientific publications can be found in the survey by Brandstätter et al. (2016).

Boyacı, Zografos, and Geroliminis (2015) describe a bi-objective mixed integer linear programming model for finding optimal locations and sizes of charging stations, as well as the size and initial allocation of the vehicle fleet for an electric car sharing system. In addition to these strategic decisions, their model optimizes the operational aspect of relocating cars throughout the planning period to mitigate problems arising from unbalanced demand. The authors then evaluate their models on real-world instances based on car sharing data from the Nice region. A major drawback of their approach is the fact that the vehicles' charge state is not considered – instead, necessary charging pauses must be provided as an input. Cepolina and Farina (2012) develop a simulated annealing algorithm to optimize the fleet size and allocation in an electric car sharing system in downtown Genoa, whereas a simulation study by Barth and Todd (1999) focuses on analyzing the effects of the trip-to-vehicle ratio on customer waiting times and the number of required relocations in a resort in southern California. In contrast to our setting, both of these publications do not consider the locations and sizes of charging stations within the network.

Several optimization problems closely related to those arising in the context of electric car sharing systems have been considered in the scientific literature. These include, for instance, problem variants considering the placement of charging stations for privately owned electric cars, for which both exact (see, Frade et al. (2011), Wang and Lin (2013), Baouche et al. (2014), Chen et al. (2013), Cavadas, Correia, and Gouveia (2015)) and heuristic (see, Wang et al. (2010), Ge, Feng, and Liu (2011), Hess et al. (2012)) methods have been proposed. An integrated model that includes the optimization of vehicle routes is described by Worley, Klabjan, and Sweda (2012). However, given that the traffic patterns of privately owned cars are necessarily quite different from those of shared cars (since private cars tend to remain parked for long durations, see Shaheen, Sperling, and Wagner (1998)), the applicability of their solution approaches to car sharing is not quite clear. Another related problem is that of finding good locations for charging stations for electric taxis, which is studied by Sellmair and Hamacher (2014), as well as by Asamer et al. (2016). While taxi traffic is very similar to one-way car sharing traffic, the fact that taxis can easily be relocated at any time again fundamentally changes the underlying model.

The problem of placing stations in a non-electric car sharing system is tackled by Correia and Antunes (2012), Correia, Jorge, and Antunes (2014) and by Fassi, Awasthi, and Viviani (2012). By the nature of that problem, any constraints related to the remaining range of a vehicle (represented by car battery levels for electric cars) are disregarded.

Other strategic optimization problems that can be of interest for electric car sharing systems include the design of the system's area of operation. This problem has, for instance, been considered by He et al. (2017), who define an algorithm based on a mixed-integer second-order cone programming formulation and evaluate it on a real-world instance from San Diego.

2 Problem definition

The charging station location problem (CSLP) considered in this article is formally defined on an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$, which models the street network of the operational area. Each edge $e \in \mathcal{E}$ has an associated length $\ell_e \geq 0$. Vertex subset $S \subseteq \mathcal{V}$ describes the set of potential locations for charging stations (simply referred to as "stations" from here on out), where each such station $i \in S$ has an associated opening cost $F_i \geq 0$, a maximum capacity (or size) $C_i \in \mathbb{N}$, and a cost per charger $Q_i \geq 0$. Furthermore, for each station $i \in S$, node set $\mathcal{N}_i \subseteq \mathcal{V}$ describes the set of nodes within walking distance, i.e., the set of nodes from / to which a user would be willing to walk to / from that station. Based on this, we define the set of nearby stations $N(v) = \{i \in S \mid v \in \mathcal{N}_i\}$ for each $v \in \mathcal{V}$.

Parameter $H \in \mathbb{N}$ defines the number of (identical) electric cars that are available for purchase. Each car has an associated acquisition cost of F_c, a uniform recharge rate per time period of ρ and a maximum battery capacity of $B^{\max}$. The total expenses for opening charging stations, building chargers and purchasing cars are limited by the budget W.

Finally, K is the set of trips potentially requested by customers, which represents an estimation of the demand for car sharing within the operational area throughout the planning period $T = \{0, \ldots, T_{\max}\}$. Each trip $k \in K$ has an associated origin $o_k \in \mathcal{V}$, a destination $d_k \in \mathcal{V}$, a start time $s_k \in T$, an end time $e_k \in T$ (where $e_k > s_k$), an (overestimated) battery consumption $b_k \in \{0, \ldots, B^{\max}\}$ and an expected profit $p_k \in \mathbb{N}$. We also define the duration of a trip as $\Delta_k = e_k - s_k$, as well as its potential start and end stations $N(o_k)$ and $N(d_k)$ according to the definition above. An exemplary instance to the CSLP is given in Figure 1a.

Feasible solutions. A solution of the CSLP is a tuple $(S', \boldsymbol{C'}, H', K', \mathcal{H}')$ consisting of a set of stations $S' \subseteq S$ that are opened, an assigned capacity (or size) $C'_i \in \{0, 1, \ldots, C_i\}$ for each station $i \in S$ (opened stations in S' have $C'_i > 0$), a number of cars $H' \leq H$ that are purchased, and a set of trips $K' \subseteq K$. For each selected trip $k \in K'$, the triple $(\mathrm{car}(k), \mathrm{start}(k), \mathrm{end}(k))$ contains its assigned car $\mathrm{car}(k) \in \{1, 2, \ldots, H'\}$ as well as the chosen start and end stations $\mathrm{start}(k) \in S' \cap N(o_k)$ and $\mathrm{end}(k) \in S' \cap N(d_k)$, respectively, and the set $\mathcal{H}'$ is defined as $\mathcal{H}' = \{(\mathrm{car}(k), \mathrm{start}(k), \mathrm{end}(k)) \mid k \in K'\}$.

This information allows to partition the set of accepted trips K' into H' pairwise disjoint non-empty trip sequences $\{K'_1, \ldots, K'_{H'}\}$ (each of which is sorted by its trips' start times) corresponding to the assignment of accepted trips to purchased cars.

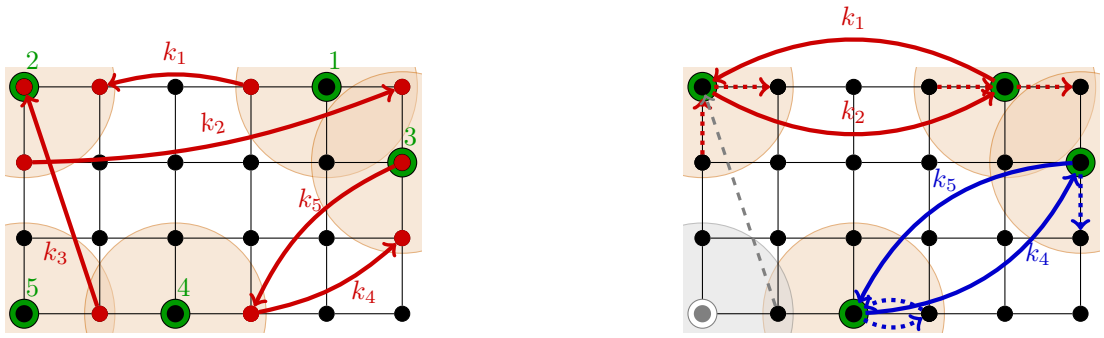
In order to be feasible, a solution must ensure that (a) its strategic expenditures do not exceed the available *budget* W, (b) the trips assigned to each car $h \in \{1, 2, \ldots, H'\}$ define a *connected route* through open stations of the street network , (c) together, all cars in the solution never exceed the available *station capacities* (at each point of time), and (d) the route assigned to each car is *battery-feasible*. A solution to the instance given in Figure 1a is given in Figure 1b.

Formally, a solution is *budget-feasible* if the expenditures for opening stations, building chargers and buying cars do not exceed the available budget W, i.e., if the inequality $\sum_{i \in S'} (F_i + Q_i C'_i) + H' F_\mathrm{c} \leq W$ holds.

To determine whether an assignment of trips to a car defines a *feasible connected route*, we need to look at each pair of consecutive trips in that car's trip sequence. Let therefore $K_h = (k_1, \ldots, k_m)$, be the sequence of trips assigned to car h in ascending order of their start time, i.e., $s_{k_j} \leq s_{k_{j+1}}$, $1 \leq j < m$. A solution is *route-feasible* if the trip sequence K_h of each purchased car $h \in \{1, 2, \ldots, H'\}$ satisfies the following two criteria: (a) no pair of two consecutive trips may be temporally overlapping, i.e., $s_{k_{j+1}} \geq e_{k_j}$, $1 \leq j < m$, and (b) the end and start stations of each such pair of consecutive trips must coincide, i.e., $\mathrm{end}(k_j) = \mathrm{start}(k_{j+1})$, $1 \leq i < m$.

A route-feasible solution is furthermore *capacity-feasible* if the number of cars that are at a station at any point in time does not exceed that station's assigned capacity C'_i. We consider car h to be (a) *at* its first trip's assigned start station $\mathrm{start}(k_1)$ from the beginning of the planning horizon 0 until the start of the first trip s_{k_1} (inclusive, in each case), (b) *at* station $\mathrm{end}(k_j) = \mathrm{start}(k_{j+1})$ from e_{k_j} until $s_{k_{j+1}}$ (again, inclusive in each case) between any two consecutive trips k_j and k_{j+1}, $1 \leq j < m$, and (c) *at* its last trip's assigned end station $\mathrm{end}(k_m)$ from that trip's end e_{k_m} until the end of the planning period $T_{\max}$ (again, inclusive in each case).

Finally, a solution is *battery-feasible* if the charge state of each car never drops below 0 when performing its assigned trips. To determine whether a car's route is battery-feasible, we need to look at its charge state at the end of each of its assigned trips. Formally, car h's charge state at the *start* of its first trip is defined as $B^{\max}$, its charge state $b^{\mathrm{s}}_{k_j}$ at the *start* of a trip k_j, $2 \leq j \leq m$, is given as $b^{\mathrm{s}}_{k_j} = \min\{B^{\max}, b^{\mathrm{e}}_{k_{j-1}} + (s_{k_j} - e_{k_{j-1}})\rho\}$, i.e., its charge state increases by ρ (up to $B^{\max}$) from what it was at the end of the previous trip for each time period it is parked at a station, and its charge state $b^{\mathrm{e}}_{k_j}$ at the end of trip k_j, $1 \leq j \leq m$, is given as $b^{\mathrm{e}}_{k_j} = b^{\mathrm{s}}_{k_j} - b_{k_j}$ i.e., its charge state decreases



(a) Instance (b) Optimal solution

Figure 1: An example instance and one of its solutions with five stations and five trips. (a) The grid graph represents the input network $\mathcal{G}$ with solid circles being customer locations and double circles being potential charging stations. Shaded regions centered at charging locations represent neighborhood areas covering customers within the allowed walking distance. There are $|S| = 5$ stations with the available trips $K = \{k_1, \ldots, k_5\}$, denoted by arrows in the input graph. Trips can be represented as tuple $(N(o_k), N(d_k), s_k, e_k, b_k, p_k)$ for all $k \in K$: $(\{1\}, \{2\}, 2, 3, 20, 1)$, $(\{2\}, \{1, 3\}, 6, 8, 60, 1)$, $(\{4, 5\}, \{2\}, 2, 6, 90, 1)$, $(\{4\}, \{3\}, 3, 6, 70, 1)$, $(\{3\}, \{4\}, 8, 10, 50, 1)$. The fleet consists of 2 cars $(H = 2)$ and the planning time-horizon $T = \{0, \ldots, 12\}$. (b) Solution consists in covering four trips with two cars: the sequence of trips assigned to the first and second car is $K_1 = (k_1, k_2)$ and $K_2 = (k_4, k_5)$, resp. Trip k_3 cannot be selected due to the limited number of available cars. Dashed arcs are trips that cannot be selected, dotted arcs represent walking from/to the customer location to/from the respective charging station.

according to the trip's battery consumption. Thus, if $b^{\mathrm{e}}_{k_j} \geq 0$ for every trip k_j in each sequence $K_h = (k_1, \ldots, k_m)$, $h \in \{1, 2, \ldots, H'\}$, a solution is *battery-feasible*.

2.1 Time-expanded location graph

As can be observed from the discussion above, it can be hard to understand solutions to the CSLP without explicitly considering the time dimension of the demand. In particular, one cannot easily check for violation of the stations capacities or whether trips associated to a single car are temporally overlapping. To overcome these difficulties, we introduce the *location graph* $G = (V, A)$ which will allow us to easily keep track of the state of stations at each relevant point in time. These time-expanded graphs, which are also referred to as time-space networks, have been used in a variety of problem settings where the underlying system's state changes over time (see, e.g., Gambella et al. (2017), Shu and Song (2013)). The vertex set $V = \{r\} \cup \bigcup_{i \in S} V_i$ of this time-expanded, directed and acyclic graph consists of an artificial root vertex r and one vertex i_t for each station $i \in S$ and each time t where a trip may either start or end at station i, i.e., $V_i = \{i_t \mid \exists t \in T^+_i \cup T^-_i\}$. Here, $T^+_i = \{t \in T : \exists k \in K : i \in N(o_k), t = s_k\}$ and $T^-_i = \{t \in T : \exists k \in K : i \in N(d_k), t = e_k\}$ denote the subset of time points where a trip may start or end at station i, respectively. The arc set $A = A^{\mathrm{I}} \cup A^{\mathrm{W}} \cup A^{\mathrm{T}}$ comprises three distinct subsets: initialization arcs from set A^{I} representing the initial allocation of cars to stations at the beginning of the planning period, waiting arcs from set A^{W} representing cars being parked (and therefore recharged) at a station for some time, and trip arcs from set A^{T} representing cars performing trips and thereby traveling between stations. The set of initialization arcs $A^{\mathrm{I}} = \{(r, i_t) | i \in S,\ t = \min\{T^+_i \cup T^-_i\}\}$ consists of one arc from the root vertex to the first (w.r.t. time) vertex of each station, whereas the set of waiting arcs $A^{\mathrm{W}} = \{(i_t, i_{t'}) | i_t, i_{t'} \in V,\ t < t', \nexists i_{t''} \in V : t < t'' < t'\}$ consists of an arc from each station vertex to the next (again w.r.t. time) vertex of that station, if such a vertex exists.

The set of trip arcs $A^{\mathrm{T}} = \bigcup_{k \in K} A^{\mathrm{T}}_k$ is defined as a union of arcs induced by the individual trips: To this end, for each trip $k \in K$, we define the set of trip arcs *induced* by that trip as $A^{\mathrm{T}}_k = \{(i_t, j_{t'}) | i \in N(o_k), j \in N(d_k), t = s_k, t' = e_k\}$, which consists of an arc from each of that trip's potential start stations at its start time to each of its potential end stations at its end time. Figure 2 shows the location graph G corresponding to the instance shown in Figure 1a, as well as the mapping of the solution given in Figure 1b into graph G.

Notice that making the time dimension explicit may in principle lead to graphs whose size grows (linearly) with $T_{\max}$, i.e., such graphs may be of pseudo-polynomial size with respect to the input data. Theorem 1 shows that this is not the case for the time-expanded location graph introduced above. Moreover, this result implies that increasing the granularity of the planning period without increasing the number of trips considered in the demand forecast does not have a large impact on the size of G and therefore on the performance of algorithms that use the location graph. This is also confirmed by the results of our computational study, see Section 6. In addition, Theorem 1 whose proof is given in Appendix A.1 is crucial for our study of a special, polynomially solvable case of CSLP in Section 3.

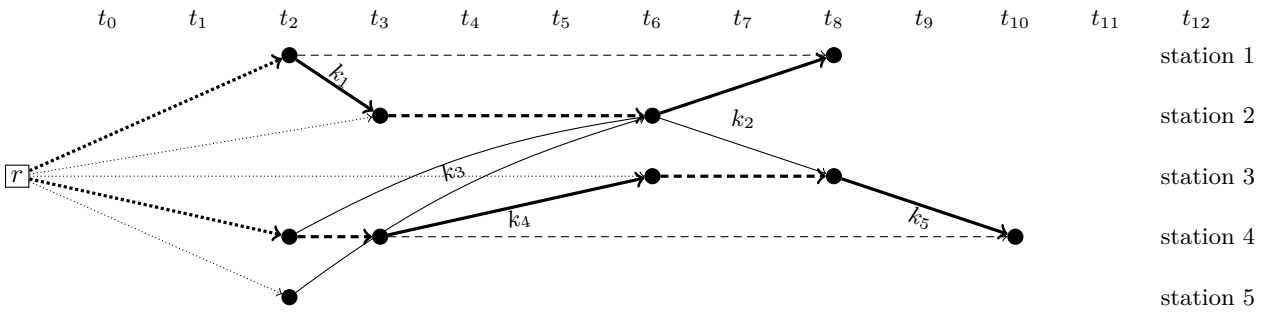


Figure 2: Location graph of the instance given in Figure 1a. Initialization arcs are dotted, waiting arcs are dashed and trip arcs are solid and labeled with their corresponding trip. The solution given in Figure 1b is highlighted in bold.

Theorem 1. *The numbers of nodes and arcs of the time-expanded location graph $G = (V, A)$ associated to an instance of CSLP can be bounded from above as follows:*

- $|V| = \mathcal{O}(\min\{|K|, |S| \cdot T_{\max}\})$ *and* $|A| = \mathcal{O}(|K|)$ *if the set of potential stations is of constant size for each node $v \in \mathcal{V}$, i.e., if $|N(v)| = \mathcal{O}(1)$ for each $v \in \mathcal{V}$.*
- $|V| = \mathcal{O}(|S| \cdot \min\{|K|, T_{\max}\})$ *and* $|A| = \mathcal{O}(|S| \cdot \min\{|K|, T_{\max}\} + |S|^2|K|)$*, otherwise.*

Observe that a trip arc does not necessarily uniquely correspond to a single trip only, but can be induced by multiple trips, as long as these trips start and end at the same times and share at least one potential start and end station. Therefore, the formulations and algorithms introduced in the following will need to distinguish between the different trips an arc can correspond to when encountering such a trip arc. Similarly, if a trip $k \in K$ allows round-trips, i.e., returning to the same station i one started from (or, more formally, if $N(o_k) \cap N(d_k) \neq \emptyset$), and no station vertices exist between i_{s_k} and i_{e_k}, algorithms must distinguish whether the arc (i_{s_k}, i_{e_k}) corresponds to performing that trip or waiting at station i for that period. This second kind of ambiguity can, in principle, be prevented by splitting the waiting arc into two arcs with another station vertex in the middle.

Another approach for dealing with these two kinds of ambiguities is allowing multiple parallel arcs between each pair of vertices, which is how our implementations handles them.

3 Problem complexity and polynomially solvable cases

In this section, we show that the CSLP as defined in Section 2 is strongly NP-hard. We furthermore prove that it remains strongly NP-hard for instances where the budget is not constraining, as well as for instances where the battery is not constraining. Finally, we show that for instances where neither the budget nor the battery is constraining, the problem is solvable in polynomial time.

First, we define what we mean by an instance that is budget- or battery-unconstrained.

Definition 1. An instance of CSLP is *budget-unconstrained* if every selection $S' \subseteq S$ of opened stations, every capacity assignment $C'_i \in \{1, \ldots, C_i\}$, $\forall i \in S'$, and every valid number $H' \leq H$ of purchased cars is feasible w.r.t. the given budget W, i.e., if $W \geq \sum_{i \in S} (F_i + C_i Q_i) + F_c H$. In that case, since opening more stations, building more chargers or buying more cars can never make a feasible solution infeasible, we can simply set $S' = S$ and $C'_i = C_i$, $\forall i \in S$. An instance that is not budget-unconstrained is *budget-constrained*.

Definition 2. An instance of CSLP is *battery-unconstrained* if any consistent assignment of trips to cars that respects the assigned station capacities is also battery-feasible. An instance that is not battery-unconstrained is *battery-constrained*.

Theorem 2. *CSLP is strongly NP-hard in general and remains strongly NP-hard for instances that are:*

- *battery-constrained but budget-unconstrained, or*
- *budget-constrained but battery-unconstrained.*

For instances that are unconstrained by both battery and budget, the problem is solvable in polynomial time.

We prove first half of this theorem by using Lemmas 1 and 2. The latter part is shown in Lemma 3 in the next subsection. The proofs of these three lemmas are provided in Appendices A.2 to A.4.

Lemma 1. *CSLP is strongly NP-hard for instances that are battery-constrained, but budget-unconstrained.*

Lemma 2. *CSLP is strongly NP-hard for instances that are budget-constrained, but battery-unconstrained.*

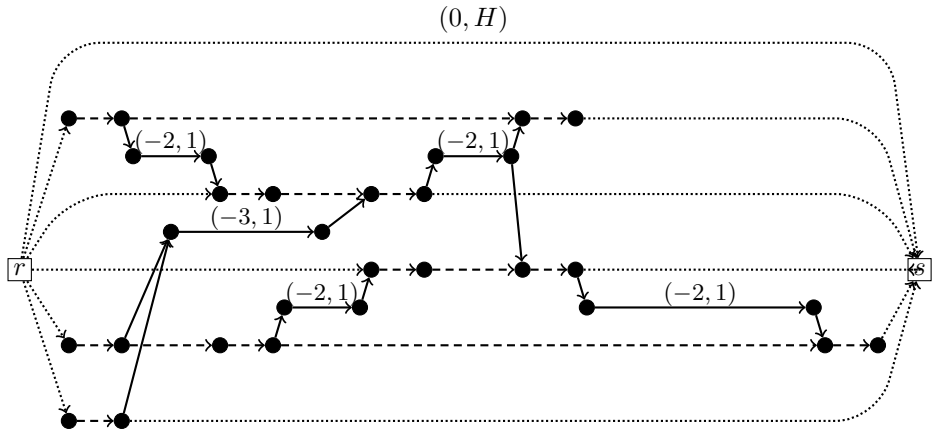


Figure 3: Flow graph of instance described in Figure 1a, arcs labeled $(profit, capacity)$. Internal root, root and sink arcs $(0, H)$ are dotted, waiting and internal station arcs $(0, C_i)$ are dashed, and internal trip arcs $(-p_k, 1)$ and start / end station arcs $(0, 1)$ are solid.

3.1 Polynomially solvable cases

We will now show that one can solve instances of the CSLP that are both budget- and battery-unconstrained in polynomial time. This result will be shown by arguing that an optimal solution to such instances can be obtained by solving a minimum-cost flow problem (with integral capacities and demands) on the *location flow network* that is obtained from the location graph via a transformation of polynomial size. The result thus immediately follows due to Theorem 1, i.e., from the fact that the size of the location graph is strictly polynomial w.r.t. the size of a considered instance.

Formally, the location flow network is defined by the time-expanded, directed, acyclic graph $\tilde{G} = (\tilde{V}, \tilde{A})$ where each arc $a \in \tilde{A}$ has an associated cost $c_a \in \mathbb{Z}$ and capacity $\varsigma_a \in \mathbb{N}$ while each node $v \in \tilde{V}$ has a (positive or negative) demand $b_v \in \mathbb{Z}$. Its node set $\tilde{V}$ is the union of station nodes $\tilde{V}^{\mathrm{S}}$ and trip nodes $\tilde{V}^{\mathrm{T}}$. Node set $\tilde{V}^{\mathrm{S}}$ is obtained from location graph G, by splitting each vertex $v \in V$ different from the root node r into two vertices $v, v' \in V'$ adopting the root node r, and adding a sink node s, i.e., $\tilde{V}^{\mathrm{S}} = \{r, s\} \cup \{i_t, i'_t \mid i_t \in V\}$. For each trip $k \in K$, two vertices v_k, v'_k are included in the set of trip nodes, i.e., $\tilde{V}^{\mathrm{T}} = \{v_k, v'_k \mid k \in K\}$. The supply of the root r as well as the demand of the sink s are equal to the number of cars ($b_r = H$, $b_s = -H$) while the demands of all other nodes are equal to zero.

The set of arcs $\tilde{A}$ is obtained as follows: Each pair of station vertices i_t, i'_t corresponding to the same node i_t in the location graph is connected by an internal station arc $a = (i_t, i'_t)$ with zero cost c_a and capacity $\varsigma_a = C_i$ equal to the maximum capacity of the corresponding station. The root r is connected to the sink s by a similar internal root arc $a = (r, s)$ with zero cost c_a and capacity $\varsigma_a = H$ equal to the maximum number of cars available for purchase. One root arc $a = (r, i_t)$ with zero arc cost c_a and capacity $\varsigma_a = H$ is added to the first vertex (w.r.t. time) of each station, as is a sink arc $a' = (i_{t'}, s)$ from each station's last vertex to the sink with the same zero cost $c_{a'}$ and capacity $\varsigma_{a'} = H$. Likewise, we add a new waiting arc $a = (i'_t, i_{t'})$ with cost $c_a = 0$ and capacity $\varsigma_a = C_i$ for each original waiting arc $(i_t, i_{t'}) \in A^{\mathrm{W}}$. For each trip $k \in K$, the corresponding pair of trip nodes v_k, v'_k is connected by an internal trip arc $a = (v_k, v'_k)$ with cost $c_a = -p_k$ and capacity $\varsigma_a = 1$, thus ensuring that the profit of each trip (modeled as negative arc cost) can be earned at most once. For each possible start node $i \in N(o_k)$ and each possible end node $j \in N(d_k)$ of trip $k \in K$, start and end station arcs (i'_{s_k}, v_k) and (v'_k, j_{e_k}) with zero costs and capacity one are included. Figure 3 shows the location flow network $\tilde{G}$ corresponding to the instance described in Figure 1a, which is obtained from the location graph given in Figure 2.

Lemma 3. *CSLP is solvable in polynomial time for instances that are both budget- and battery-unconstrained.*

4 Integer linear programming formulations

We now describe a generic ILP formulation for the CSLP which is given by (1)–(12). Variables $y_i \in \{0, 1\}$, $\forall i \in S$, indicate whether station i was opened or not, whereas variables $z_i \in \{0, \ldots, C_i\}$ indicate the number of chargers built at that station (i.e., its *size* or *capacity*). Likewise, variables $a_h \in \{0, 1\}$, $\forall h \in \{1, 2, \ldots, H\}$, indicate whether car h was purchased or not. Finally, variables $x_k \in \{0, 1\}, \forall k \in K$, determine whether trip k has been accepted or not, and variables $x_k^h \in \{0, 1\}$, $\forall k \in K$, $\forall h \in \{1, 2, \ldots, H\}$, specify whether a trip k has been assigned to car h.

$$\max \sum_{k \in K} p_k x_k \tag{1}$$

$$\text{s.t.} \sum_{i \in S} (F_i y_i + Q_i z_i) + \sum_{h=1}^{H} F_c a_h \leq W \tag{2}$$

$$z_i \leq C_i y_i \qquad \forall i \in S \tag{3}$$

$$\sum_{h=1}^{H} x_k^h = x_k \qquad \forall k \in K \tag{4}$$

$$\sum_{k \in K: s_k \leq t, e_k > t} x_k^h \leq a_h \qquad \forall h \in \{1, 2, \ldots, H\}, \forall t \in \bigcup_{i \in S} T_i^+ \tag{5}$$

$$(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{a}) \in \mathcal{L} \tag{6}$$

$$(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{a}) \in \mathcal{B} \tag{7}$$

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

$$z_i \in \{0, \ldots, C_i\} \qquad \forall i \in S \tag{9}$$

$$a_h \in \{0, 1\} \qquad \forall h \in \{1, 2, \ldots, H\} \tag{10}$$

$$x_k \in \{0, 1\} \qquad \forall k \in K \tag{11}$$

$$x_k^h \in \{0, 1\} \qquad \forall h \in \{1, 2, \ldots, H\}, \forall k \in K \tag{12}$$

The objective function (1) maximizes the total profit gained from accepting trips, while the budget constraint (2) ensures that the expenditures related to the opening of stations, the construction of chargers and the purchase of cars do not exceed the given budget W. Inequalities (3) ensure that the assigned capacity of an opened station never exceeds its maximum capacity, while simultaneously ensuring that no chargers are built at stations that are not opened. Equations (4) assign a car to each trip that was accepted, while inequalities (5) ensure that at most one trip from each set of temporally overlapping trips can be assigned to a car. They also ensure that a car needs to be acquired if at least one trip is assigned to it. Note that it is not necessary to impose constraints (5) for all time points in the planning period. Instead, it is sufficient to consider only those times t at which new trips may start, cf. notation T_i^+ introduced in Section 2.1.

Abstract constraint (6) considers the set $\mathcal{L}$ of all incidence vectors of $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{a})$ that correspond to consistent car movements through the station network. They ensure that assignment of trips to each car defines a *feasible connected route* that respects station capacities (see introduction). Concrete realizations of this constraint will be given in Section 4.1.

In a similar way, constraint (7) considers the set $\mathcal{B}$ of all incidence vectors of $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{a})$ that correspond to battery feasible assignments of trips to cars. Thus, it is ensured that the battery of any car will not be depleted below its minimum allowed charge state (which we refer to as zero) when that car performs all trips assigned to it. Concrete realizations constraints (7) will be given in Section 4.2.

Finally, constraints (8) – (12) enforce integrality on the decision variables and ensure that they remain within their associated bounds.

4.1 Enforcing consistent car movement

In this subsection, we provide two possibilities for replacing constraint (6) by integer linear programs: one based on a multi-commodity flow formulation in the location graph and one based on the explicit assignment of a start and end station to each accepted trip.

4.1.1 Multi-commodity flow model

The multi-commodity flow formulation (13)–(18) uses the location graph introduced in Section 2.1 to represent the movement of cars through the station network during the planning period. Flow variables $f_a^h \in \{0, 1\}$, $\forall a \in A$, $\forall h \in \{1, 2, \ldots, H\}$, indicate whether car h moves along a specific arc a during the planning period. Such movement can be both spatial and temporal (in case of trip arcs), or only temporal (in case of waiting arcs). Flow f_a^h on an initialization arc $(r, i_t) \in A^{\mathrm{I}}$ models that car h starts at station i at the start of the planning period.

$$\sum_{h=1}^{H} f^h[\delta^-(i_t)] \leq z_i \qquad \forall i \in S, \forall t \in \left(T_i^- \cup \{t' \mid \exists (r, i_{t'}) \in A\}\right) \tag{13}$$

$$f^h[\delta^+(r)] = a_h \qquad \forall h \in \{1, 2, \ldots, H\} \tag{14}$$

$$f^h[\delta^-(i_t)] - f^h[\delta^+(i_t)] = 0 \qquad \forall h \in \{1, 2, \ldots, H\}, \forall i_t \in V, \exists (i_t, j_{t'}) \in A : t' > t \tag{15}$$

$$f^h[\delta^-(i_t)] - f^h[\delta^+(i_t)] \geq 0 \qquad \forall h \in \{1, 2, \ldots, H\},\ \forall i \in S,\ t = \max(T_i^+ \cup T_i^-) \tag{16}$$

$$\sum_{a \in A^k_{\mathrm{T}}} f^h_a = x^h_k \qquad \forall h \in \{1,2,\dots,H\},\ \forall k \in K \tag{17}$$

$$f^h_a \in \{0,1\} \qquad \forall h \in \{1,2,\dots,H\},\ \forall a \in A \tag{18}$$

Inequalities (13) ensure that at each point in time, only as many cars enter a station as there are chargers available at that station. Note that parked cars from the previous period "reenter" the station via the waiting arcs. Since the number of cars parked at a station can only increase by arriving trips and by initially allocating cars to that station, it is sufficient to impose these constraints for each station $i \in S$ and for all times $t \in T_i^- \cup \{t' \mid \exists (r, i_{t'}) \in A\}$. These constraints also guarantee that no car may enter a station that has not been opened (since its assigned capacity must be zero, due to constraints (3)). Together, equations (14), (15), and (16) ensure that for each car that was purchased, one unit of flow leaves the root and is routed through the whole network. Inequalities (16) rather than equations are used for the last node of each station to account for cars parked at that station until the end of the planning horizon. The slack of constraints (16) will correspond to the number of cars parked at a station $i \in S$ from time $\max\left(T_i^+ \cup T_i^-\right)$ until time $T_{\max}$. Finally, linking constraints (17) ensure that for each trip assigned to a specific car, that car's corresponding flow is routed over one of the trip's travel arcs.

Besides the design variables introduced in the previous subsection, this compact model requires additional $\mathcal{O}(H \cdot |A|)$ variables and $\mathcal{O}(H \cdot \max\{|V|, |K|\})$ constraints to ensure feasibility with respect to consistent car movements.

4.1.2 Connectivity cut model

In the following, we introduce an alternative model to enforce consistent car movement. It utilizes the following additional variables to keep track of each car's location throughout the planning period: For each trip $k \in K$ and each station $i \in N(o_k)$, variable $x^{\mathrm{out}}_{ki} \in \{0,1\}$ indicates whether trip k starts at station i while variable $x^{\mathrm{in}}_{ki} \in \{0,1\}$ indicates, for each station $i \in N(d_k)$, whether that trip ends at station i. Furthermore, variables $a^{\mathrm{out}}_{hi} \in \{0,1\}$, $\forall h \in \{1,2,\dots,H\}$, $\forall i \in S$, define whether car h starts at station i at the beginning of the planning period.

A valid model for enforcing consistent car movement is obtained by replacing abstract constraints (6) by constraints (19)–(30) in the generic ILP formulation introduced above.

$$\sum_{i \in N(o_k)} x^{\mathrm{out}}_{ki} = x_k \qquad \forall k \in K \tag{19}$$

$$\sum_{i \in N(d_k)} x^{\mathrm{in}}_{ki} = x_k \qquad \forall k \in K \tag{20}$$

$$x^{\mathrm{out}}_{ki} \le y_i \qquad \forall k \in K,\ \forall i \in N(o_k) \tag{21}$$

$$x^{\mathrm{in}}_{ki} \le y_i \qquad \forall k \in K,\ \forall i \in N(d_k) \tag{22}$$

$$\sum_{i \in S} a^{\mathrm{out}}_{hi} = a_h \qquad \forall h \in \{1,2,\dots,H\} \tag{23}$$

$$a^{\mathrm{out}}_{hi} \le y_i \qquad \forall i \in S,\ \forall h \in \{1,2,\dots,H\} \tag{24}$$

$$\sum_{h=1}^{H} a^{\mathrm{out}}_{hi} - \sum_{\substack{k \in K: s_k < t, \\ i \in N(o_k)}} x^{\mathrm{out}}_{ki} + \sum_{\substack{k \in K: e_k \le t, \\ i \in N(d_k)}} x^{\mathrm{in}}_{ki} \le z_i \qquad \forall i \in S,\ \forall t \in T_i^- \cup \{\min(T_i^- \cup T_i^+)\} \tag{25}$$

$$\sum_{h=1}^{H} a^{\mathrm{out}}_{hi} - \sum_{\substack{k \in K: s_k < t, \\ i \in N(o_k)}} x^{\mathrm{out}}_{ki} + \sum_{\substack{k \in K: e_k \le t, \\ i \in N(d_k)}} x^{\mathrm{in}}_{ki} \ge \sum_{\substack{k \in K: i \in N(o_k), \\ s_k = t}} x^{\mathrm{out}}_{ki} \qquad \forall i \in S,\ \forall t \in T_i^+ \tag{26}$$

$$(1 - x^h_{k_1}) + (1 - x^h_{k_2}) + (1 - x^{\mathrm{in}}_{k_1 i_1}) + (1 - x^{\mathrm{out}}_{k_2 i_2}) + \sum_{\substack{k \in K: s_k \ge e_{k_1}, \\ e_k \le s_{k_2}, o_k \in N(i_1)}} x^h_k \ge 1 \qquad \begin{array}{r} \forall h \in \{1,2,\dots,H\},\ \forall k_1, k_2 \in K, \\ \forall i_1 \in N(d_{k_1}),\ \forall i_2 \in N(o_{k_2}), \\ e_{k_1} \le s_{k_2},\ i_1 \ne i_2 \end{array} \tag{27}$$

$$x^{\mathrm{out}}_{ki} \in \{0,1\} \qquad \forall k \in K, \forall i \in N(o_k) \tag{28}$$

$$x^{\mathrm{in}}_{ki} \in \{0,1\} \qquad \forall k \in K, \forall i \in N(d_k) \tag{29}$$

$$a^{\mathrm{out}}_{hi} \in \{0,1\} \qquad \forall h \in \{1,2,\dots,H\}, \forall i \in S \tag{30}$$

Equations (19) and (20) ensure that each selected trip is assigned a start and end station from its respective neighborhoods, while inequalities (21) and (22) ensure that only opened stations may be used as start or end station of accepted trips. Likewise, equations (23) and inequalities (24) ensure that each car is assigned an opened station from which it starts. Inequalities (25) guarantee that the number of cars parked at a station at any time never exceeds that station's assigned capacity. Note that the left hand side (LHS) is simply the number of cars parked at station i at

time t. The latter is obtained as the sum of cars initially starting at station i minus the sum of cars leaving station i earlier than t plus the sum of cars arriving at i not later than t. These constraints are imposed only for those points in time at which the number of cars at station i may increase, i.e., at the first relevant time point (initial car allocation) and whenever a trip may arrive. Inequalities (26) ensure that the number of cars leaving station i at time t does not exceed the available number of cars. Observe that the LHS of constraints (26) is identical to the LHS of previously explained constraints (25) and that the right hand side (RHS) of inequalities (26) is the number of cars leaving node starting a trip from station i at time t. These constraints are considered for all time points at which the number of parked cars may decrease, i.e., whenever a trip may start from a station. Constraints (27) are added dynamically whenever we find a solution with inconsistent car locations (i.e., a car is assigned two successive trips k_1 and k_2, but k_2 starts at a station that is different from the one where k_1 ends. In that case, either (a) one of the trips must not be assigned to that car (or not performed altogether), or (b) the end station of k_1 or the start station of k_2 must change, or (c) at least one more trip that can start at i_1 and that lies completely within the time window between k_1 and k_2 must be assigned to that car. These constraints could be strengthened by only considering those trips in (c) for which a battery- and location-feasible path between i_1 and i_2 can exist.

Observe the following connection with the flow-variables of the multi-commodity flow model:

$$f^h_{(r,i_t)} = 1, \text{ for some } (r, i_t) \in A^{\mathrm{I}} \quad \Leftrightarrow \quad a^{\mathrm{out}}_{hi} = 1 \tag{31}$$

$$f^h_{(i_t,j_{t'})} = 1, \text{ for some } (i_t, j_{t'}) \in A^{\mathrm{T}} \quad \Leftrightarrow \quad x^{\mathrm{out}}_{ki} = 1, x^{\mathrm{in}}_{kj} = 1, \tag{32}$$

This connectivity cut based model comprises additional $\mathcal{O}(|K| \cdot |S|)$ decision variables (under the reasonable assumption that $H < |K|$) and $\mathcal{O}(H \cdot |K|^2 \cdot |S|)$ constraints. Notice, however, that without constraints (27), only $\mathcal{O}(|S| \cdot \max\{T_{max}, |K|\})$ constraints are required to initialize the model, whereas the remaining ones are dynamically separated, which means that the number of violated connectivity constraints is significantly below its theoretical upper bound. Hence, assuming that the number of potential stations is rather limited, this formulation may exhibit a computational advantage when compared to the flow-based model due to the size of the underlying LP-relaxation

Valid inequalities. In the following, we introduce a family of valid inequalities that can be used to strengthen the LP-relaxation and that can be separated on-the-fly. For each trip $k \in K$, let $N^+(k)$ be defined as the set of trips ending before s_k that, if performed by the same car, *can newly enable* that car to perform trip k by potentially moving from a station that is not one of its start stations $N(o_k)$ to one that is. Formally, this set is defined as

$$N^+(k) = \{k' \in K : e_{k'} \leq s_k, N(o_{k'}) \setminus N(o_k) \neq \emptyset, N(d_{k'}) \cap N(o_k) \neq \emptyset\}.$$

Similarly, for each trip $k \in K$, set $N^-(k)$ is the set of trips starting before s_k that, if performed by the same car, make it impossible for that car to perform trip k as immediate successor by always going from one of its start stations $N(o_k)$ to one that is not, i.e., we have

$$N^-(k) = \{k' \in K : s_{k'} \leq s_k, N(o_{k'}) \subseteq N(o_k), N(d_{k'}) \subseteq S \setminus N(o_k)\}.$$

Theorem 3. *The following inequalities are valid for the connectivity cut formulation:*

$$x^h_k \leq \sum_{i \in N(o_k)} a^{\mathrm{out}}_{hi} + \sum_{k' \in N^+(k)} x^h_{k'} - \sum_{k' \in N^-(k)} x^h_{k'} \qquad \forall k \in K, \forall h \in \{1, 2, \ldots, H\} \tag{33}$$

Proof. Validity of these constraints follows from the following argument: observe that a trip k can only be assigned to a car if that car is potentially in one of k's possible start stations $N(o_k)$ at s_k (i.e., the times it potentially enters them minus the times it certainly leaves them must be at least one). A car potentially enters that set of stations if it (a) starts there (= first term) or if it (b) performs a trip that can start outside and end inside (no later than s_k) (= second term). Likewise, it certainly leaves the set if it performs a trip that must start inside (no later than s_k) and end outside of it (= third term). □

Finally, observe that if $|N(o_k)| = |N(d_k)| = 1, \ \forall k \in K$, these constraints alone already ensure connectivity, thus no connectivity cuts (27) must be added.

4.2 Considering car battery charge states

In this subsection, we propose three possible ILP formulations to model constraint (7) that ensure feasibility of a solution with respect to battery consumption. Observe that it is sufficient to track each cars battery state at the beginning and end of all trips it may possibly perform. We therefore consider the sets of start and end times of trips $T^+ = \bigcup_{i \in S} T^+_i$ and $T^- = \bigcup_{i \in S} T^-_i$, respectively. Let furthermore $T^{\mathrm{B}} = \{0\} \cup T^+ \cup T^-$ be the set of battery relevant time points and let $\pi(t) = \max\{t' \in T^{\mathrm{B}} \mid t' < t\}$ denote the preceding time point from T^{B} for each $t \in T^{\mathrm{B}} \setminus \{0\}$. We also observe that $T^{\mathrm{B}} \in \mathcal{O}(\min\{|K|, T_{\max}\})$ and recall that we assume that all cars are fully charged in the beginning (i.e., at time $t = 0$).

4.2.1 Battery tracking using continuous variables

Our first approach is based on the observation that we can track each car's battery state by using continuous variables $g_t^h \in [0, B^{\max}]$, $\forall h \in \{1, 2, \ldots, H\}$, $\forall t \in T^{\mathrm{B}}$, which indicate the charge state of car h's battery at time t. The following constraints can be used to replace (7) in this case:

$$g_{e_k}^h - g_{s_k}^h \leq -b_k x_k^h + \Delta_k \rho (1 - x_k^h) \qquad \forall h \in \{1, 2, \ldots, H\},\ \forall k \in K \tag{34}$$

$$g_t^h - g_{\pi(t)}^h \leq \rho(t - \pi(t)) a_h \qquad \forall h \in \{1, 2, \ldots, H\},\ \forall t \in T^{\mathrm{B}} \setminus \{0\} \tag{35}$$

$$g_0^h = B^{\max} \qquad \forall h \in \{1, 2, \ldots, H\} \tag{36}$$

$$0 \leq g_t^h \leq B^{\max} \qquad \forall h \in \{1, 2, \ldots, H\},\ \forall t \in T^{\mathrm{B}} \setminus \{0\} \tag{37}$$

Inequalities (34) ensure that if and only if a trip k is assigned to a car h (i.e., $x_k^h = 1$), that car's battery is depleted according to the trip's associated energy consumption b_k. Otherwise, the constant $\Delta_k \rho$ (which corresponds to the maximum battery charging between s_k and e_k) is used to deactivate the constraint. Inequalities (35) limit the recharging rate of each car to ρ per unit of time. Equations (36) simply initialize all cars to have a fully charged battery at the start. Finally, inequalities (37) ensure that a car's battery level never drops below zero or exceeds its maximum capacity. This model requires $\mathcal{O}(H \cdot T^{\mathrm{B}}) = \mathcal{O}(H \cdot \min\{|K|, T_{\max}\})$ variables and $\mathcal{O}(H \cdot \max\{|K|, T^{\mathrm{B}}\}) = \mathcal{O}(H \cdot |K|)$ constraints to guarantee feasibility of the solution with respect to the battery consumption.

4.2.2 Battery-infeasible path cuts

An alternative approach that assures feasibility of every assignment of trips to a car is based on the idea of explicitly forbidding infeasible assignments. To that end, we add constraints

$$\sum_{k \in \bar{K}} x_k^h \leq \mathcal{M}_{\bar{K}} a_h \qquad \forall \bar{K} \subseteq K,\ \forall h \in \{1, 2, \ldots, H\} \tag{38}$$

to our model, where $\mathcal{M}_{\bar{K}}$ is the maximum number of trips from set $\bar{K}$ that can be performed by any car without falling below its minimum battery capacity. Observe that in this case, no additional variables need to be introduced, but this comes at the price of exponentially many infeasible path constraints that need to be added dynamically in a branch-and-cut procedure.

4.2.3 Time-expanded battery graph

Finally, in this third model, we use a similar construction as in Section 2.1 to keep track of each car's charge state throughout the planning period. The directed time-expanded *battery graph* $G_{\mathrm{B}} = (V_{\mathrm{B}}, A_{\mathrm{B}})$ contains nodes and arcs according to all possible battery states of a car at each relevant time point of the planning horizon. Nodes of the set V_B, denoted by u_t, represent a battery state of u at time t, where $u \in \{0, \ldots, B^{\max}\}$ and $t \in T^{\mathrm{B}}$. Nodes in V_B are connected by arcs that correspond to charging the battery, or taking a trip (in which case the battery is depleted). Arcs $a = (u_t, v_{t'}) \in A_{\mathrm{B}}$ represent a possible change of the battery level from u at time t to v at time t'.

Since it is not necessary to consider all possible combinations of u and t, we now define the time-expanded battery graph recursively. Thus, let $G_{\mathrm{B}}(t) = (V_{\mathrm{B}}(t), A_{\mathrm{B}}(t))$ be the subgraph of G_{B} induced by all vertices $\{u_{t'} | t' \leq t\}$ for each $t \in T^{\mathrm{B}}$.

$$\begin{aligned}
V_{\mathrm{B}}(0) &= \{B_0^{\max}\} \\
V_{\mathrm{B}}(t) &= V_{\mathrm{B}}(\pi(t)) \\
&\quad \cup \{v_t \mid v = \min\{u + (t - \pi(t))\rho,\ B^{\max}\},\ u_{\pi(t)} \in V_{\mathrm{B}}(\pi(t))\} \\
&\quad \cup \{u_{e_k} \mid k \in K,\ (u + b_k)_{s_k} \in V_{\mathrm{B}}(s_k),\ u \geq 0\} \\
A_{\mathrm{B}}(0) &= \emptyset \\
A_{\mathrm{B}}(t) &= A_{\mathrm{B}}(\pi(t)) \\
&\quad \cup \{(u_{\pi(t)}, v_t) \mid v = \min\{u + (t - \pi(t))\rho,\ B^{\max}\},\ u_{\pi(t)} \in V_{\mathrm{B}}(\pi(t))\} \\
&\quad \cup \{((u + b_k)_{s_k}, u_{e_k}) \mid k \in K,\ (u + b_k)_{s_k} \in V_{\mathrm{B}}(s_k)\}
\end{aligned}$$

We denote the set of depletion arcs induced by trip k by $A_{\mathrm{B}}^k = \{((u + b_k)_{s_k}, u_{e_k}) \mid (u + b_k)_{s_k} \in V_{\mathrm{B}}\}$. Figure 4 shows the battery graph corresponding to the instance given in Figure 1a. Note that while that graph may, in principle, grow quite large, its size can be drastically reduced by rounding up all battery consumptions to a fixed number of

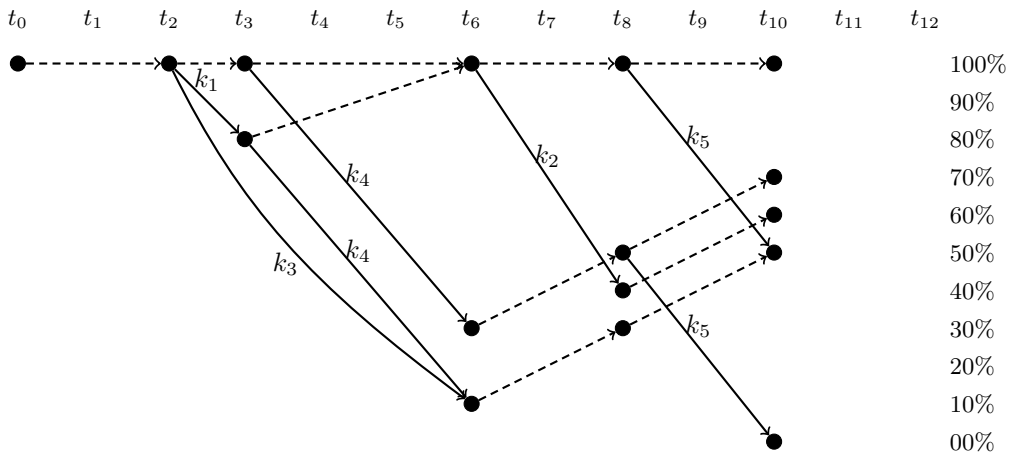


Figure 4: Battery graph of the instance given in Figure 1a. Charging arcs are dashed, depletion arcs are solid and labeled with their corresponding trip.

potential consumptions. While this may result in some optimal solutions being discarded, the resulting solution is always feasible for the original problem. [1]

We keep track of the individual cars' battery states by using flow variables $g_a^h \in \{0,1\}$, $\forall h \in \{1,2,\dots,H\}$, $\forall a = (u_t, v_{t'}) \in A_{\mathrm{B}}$, which are equal to one if and only if car h's battery level changes from u at time t to v at time t'.

$$g^h[\delta^+(B_0^{\max})] = a_h \qquad \forall h \in \{1,2,\dots,H\} \tag{39}$$

$$g^h[\delta^-(u_t)] - g^h[\delta^+(u_t)] = 0 \qquad \forall h \in \{1,2,\dots,H\},\ \forall u_t \in V_{\mathrm{B}},\ \exists (u_t, v_t') \in A_{\mathrm{B}} : t' > t \tag{40}$$

$$\sum_{a \in A_{\mathrm{B}}^k} g_a^h = x_k^h \qquad \forall h \in \{1,2,\dots,H\},\ \forall k \in K \tag{41}$$

$$g_a^h \in \{0,1\} \qquad \forall h \in \{1,2,\dots,H\},\ \forall a \in A_{\mathrm{B}} \tag{42}$$

Equations (39) and (40) ensure that for each car that was purchased, one unit of flow leaves the root and is routed through the whole network. Equations (41) simply require that for each trip assigned to a car, that car's battery is depleted according to the trip's consumption by routing flow over one of the arcs induced by it. Since the maximum number of nodes and arcs of G_{B} are each bounded from above by $\mathcal{O}(B^{\max} \cdot T^{\mathrm{B}}) = \mathcal{O}(B^{\max} \cdot \min\{|K|, T_{\max}\})$, this model requires $\mathcal{O}(H \cdot B^{\max} \cdot \min\{|K|, T_{\max}\})$ variables and constraints.

5 Algorithms

In this section, we describe our algorithms for separating those constraints that are added dynamically during branch-and-cut (see Section 5.1) and also detail two heuristic methods for computing feasible solution to the CSLP. The first heuristic is based on iteratively finding resource-constrained shortest paths in the location graph (see Section 5.2), and the second one is based on finding a minimum cost flow in the location flow network (see Section 5.3).

5.1 Constraint separation

As mentioned above, both formulations for ensuring consistent car movements, the flow-based and the cut-based model, are of polynomial size. However, it sometimes pays off to dynamically separate some of the model constraints, to keep the size of the underlying LP as small as possible. This is particularly important for the connectivity-cut model, which includes $\mathcal{O}(H \cdot |K|^2)$ cutset constraints (27) whose separation is detailed in Section 5.1.1. Concerning the models that ensure battery feasible car-to-trip assignments, we explain in Section 5.1.2 how we separate cuts from the exponentially large family of infeasible-path constraints (38).

5.1.1 Connectivity cut model

Our separation procedure for constraints (27) distinguishes between points of the LP-relaxation that satisfy the integrality constraints (in which case we perform the so-called *integer separation* described below), or not (in which

[1] Also note that nonlinear charging functions could be easily considered when using battery graph G_{B}. Given an initial battery state at time $\pi(t)$ and a recharging time of $t - \pi(t)$ one would simply need to replace the value $(t - \pi(t))\rho$ by the corresponding result obtained from such a general charging function in above definition of V_{B} and A_{B}.

case *fractional separation* is performed).

Integer separation. In this case, we are given an LP-solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{a}}, \bar{\boldsymbol{x}}^{in}, \bar{\boldsymbol{x}}^{out}, \bar{\boldsymbol{a}}^{out})$ that satisfies the integrality conditions of all variables. In order to find a possibly violated inequality (27), we iterate over each car's assigned trips in order of their start times (note that constraints (5) ensure that the trips assigned to a car by the current solution cannot overlap and have uniquely defined start and end stations). Whenever we find two successive trips k_1 and k_2 that are disconnected (i.e., the assigned end station of k_1 differs from the assigned start station of k_2), we add the corresponding connectivity constraint (27).

Fractional separation. In the fractional case, a major difficulty in separating the current LP-solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{a}}, \bar{\boldsymbol{x}}^{in}, \bar{\boldsymbol{x}}^{out}, \bar{\boldsymbol{a}}^{out})$ arises from the fact that a trip does not necessarily have a uniquely defined (a) successor trip, (b) start station or (c) end station. We therefore have to (potentially) consider multiple successor trips for each trip.

For a given threshold β_0 (we set $\beta_0 = 0.1$ in our computational study) and each car h, we consider all trips k_1 such that $\bar{x}^h_{k_1} \geq 2\beta_0$. For each such trip k_1, we consider each potentially succeeding trip k_2 (i.e., every trip where $s_{k_2} \geq e_{k_1}$ and $N(d_{k_1}) \cap N(o_{k_2}) \neq \emptyset$) for which the appropriate connectivity constraints may possibly be violated, i.e., if $\bar{x}^h_{k_1} + \bar{x}^h_{k_2} \geq 1$. For each such pair k_1, k_2 we consider each combination of potential end station i_1 of k_1 and start station i_2 of k_2 for which the two trips are disconnected, i.e., every $(i_1, i_2) \in N(d_{k_1}) \times N(o_{k_2})$ such that $i_1 \neq i_2$, and for which the corresponding connectivity constraint may be violated, i.e. those for which $\bar{x}^h_{k_1} + \bar{x}^h_{k_2} + \bar{x}^{\text{in}}_{k_1 i_1} + \bar{x}^{\text{out}}_{k_2 i_2} > 3$ holds. Finally, for every such k_1, k_2, i_1, i_2, we check whether $\bar{x}^h_{k_1} + \bar{x}^h_{k_2} + \bar{x}^{\text{in}}_{k_1 i_1} + \bar{x}^{\text{out}}_{k_2 i_2} > 3 + \sum_{\substack{k \in K : s_k \geq e_{k_1}, \\ e_k \leq s_{k_2}, o_k \in \mathcal{N}_{i_1}}} \bar{x}^h_k$, in which case a violated constraint (27) has been identified.

5.1.2 Battery-infeasible path cuts

As above, our separation routine for identifying violated battery-infeasible path cuts (38) distinguishes whether the current LP solution is integral or not.

Integer separation. Similar to the case of connectivity cuts, we identify violated inequalities in case the current LP solution is integral by iterating over each car's set of assigned trips in ascending order of start time while keeping track of the associated charge state. To this end, starting at the initial battery level of $B^{\max}$, we subtract b_k from the current charge state whenever the car performs a trip k and add $\rho(s_{k_2} - e_{k_1})$ (up to at most $B^{\max}$) to it for every recharging break between two successive trips k_1, k_2. A resulting charge state below zero indicates that the set of trips $\bar{K}$ performed up to this point is battery infeasible. In that case, we first aim to reduce the size of trip set $\bar{K}$ by removing trips from the front of the sequence as long as the resulting subset is still battery-infeasible and then add the corresponding battery cut $\sum_{k \in \bar{K}} x^h_k \leq |\bar{K} - 1| a_h$. To avoid computational overhead when reducing a set of trips, we immediately clear the current trip sequence whenever a car is fully recharged, i.e., when the charge state reaches $B^{\max}$.

Fractional separation. As mentioned above, the main difficulty of identifying violated inequalities when some variable values are fractional stems from the fact that there is typically not a single uniquely defined successor of each trip. Instead, several trips (assigned to the car with some value between zero and one) may succeed a given trip. To consider multiple potential paths for every car on the one hand and avoid enumerating and checking too many such paths, we again consider above mentioned threshold parameter β_0. For each car h, we first identify the set of trips $\bar{K}(h)$ that do not have any potential predecessor w.r.t. car h and parameter β_0 in the current LP solution, i.e., all trips $k \in K(h)$ with $\bar{x}^h_k \geq \beta_0$ for some car h such that there does not exist another trip $k' \in K$ with $\bar{x}^h_{k'} \geq \beta_0$ and $e_{k'} \leq s_k$. For each car h, set $\bar{K}(h)$ determines the seeds of all trip sequences (paths) that will be checked for possibly violated battery-infeasible cuts. To this end, all valid successors w.r.t. to car h and parameter β_0 of each current subpath are considered that may possibly yield a violated constraint. More precisely, for each car h, we consider the set of paths $\{(k_1, k_2, \ldots, k_n) \mid k_{i_1} \in \bar{K}(h),\ x^h_{k_i} \geq \beta_0,\ s_{k_i} \geq e_{k_{i-1}},\ N(d_{k_{i-1}}) \cap N(o_{k_i}) \neq \emptyset,\ 2 \leq i \leq n\}$ such that $\sum_{i=1}^{l} x^h_{k_i} \geq l - 1$, for all $l \in \{1, 2, \ldots, n\}$ (since the constraint cannot possibly be violated otherwise).

Whenever a (partial) path $(k_1, k_2, \ldots, k_{n-1})$ is extended by adding a new trip k_n we check the battery level before and after performing the trip (which can be easily calculated, since we always know the battery level at the end of k_{n-1}). If the charge state is $B^{\max}$ at the start of k_n, we end the current path and start a new one consisting only of k_n (since the path so far was not battery-infeasible, and any subsequently found battery-infeasible path would still be infeasible after dropping $k_1, \ldots, k_{n-1}$). If, on the other hand, the charge state is below zero at the end of k_n, we found a battery-infeasible path and add the corresponding cut after potentially removing trips from its front, as described above. If neither of those two conditions hold, we simply continue following the path until no more possible successors exist.

5.2 Path-based heuristic

As described in Section 2, a feasible solution to the CSLP comprises a set of battery-feasible paths through the location graph (or, alternatively, through the original street network). Thus, one way of heuristically computing a feasible solution is to iteratively find such paths (with high profit) and add them to a current solution while ensuring that it remains budget- and capacity-feasible. This idea is the basis of the *path-based heuristic* (PH) introduced in this section, whose pseudocode is given in Algorithm 1.

$(S', \boldsymbol{C'}, H', K', \mathcal{H}') = (\emptyset, \mathbf{0}, 0, \emptyset, \emptyset)$ // initial solution
$(V', A') = (V, A)$
$W' = W$ // remaining budget
$U_{it} = 0, \forall i \in S,\ \forall t \in T_i^+ \cup T_i^-$ // cars parked at station i at time t
$P = (\emptyset, \emptyset)$
repeat
  $P = \texttt{RCSP}((V', A'), W', \boldsymbol{C'}, \boldsymbol{U})$ // find path $P = (V(P), A(P))$ with cost $c(P)$
  if $c(P) < 0$ **then** // if path P is profitable
    $H' = H' + 1$
    $W' = W' - F_{\mathrm{c}}$
    for $i_t \in V(P)$ **do**
      $U_{it} = U_{it} + 1$
      if $i \in S \setminus S'$ **then** // build station
        $S' = S' \cup \{i\}$
        $W' = W' - F_i$
      if $C'_i < U_{it}$ **then** // build charger
        $C'_i = U_{it}$
        $W' = W' - Q_i$
    for $k \in K : \exists (i_t, j_{t'}) \in A(P) \cap A_k^{\mathrm{T}} \neq \emptyset$ **do** // all trips in path P
      $K' = K' \cup \{k\}$
      $A' = A' \setminus A_k^{\mathrm{T}}$ // remove trip arcs of newly accepted trips
      $\mathrm{car}(k) = H'$, $\mathrm{start}'(k) = i$, $\mathrm{end}'(k) = j$
    for $i_t \in V(P)$ **do** // remove incoming arcs from vertices at maximum capacity
      if $U_{it} = C_i$ **then** $A' = A' \setminus \delta^-(i_t)$
until $H' = H$ **or** $W' < F_{\mathrm{c}}$ **or** $c(P) \geq 0$

Algorithm 1: Path-based heuristic.

Algorithm 1 starts from an empty solution in which no stations are opened, no cars are purchased and no trips are accepted. It also keeps track of the remaining budget W' (which is initialized to be the budget W), and the number of cars U_{it} that enter station i at time t. Based on the so-far constructed solution and the remaining budget, a most profitable route for an additional car is identified. To this end, a battery-feasible path P through the location graph is identified that maximizes the additionally obtained profit, whose inclusion in the solution is budget-feasible and that does not violate any capacity constraints. Interpreting profits as negative costs on trip arcs, such a path can be found by identifying a *resource-constrained shortest path* (RCSP) in location graph G, see below for a detailed description.

If a path with positive profit (i.e., negative cost) is found, the current solution, the remaining budget as well as values U_{it} are updated accordingly. Trip arcs of all selected trips are removed in order to ensure that one cannot assign those trips to another car in a later iteration. Additionally, we remove all incoming arcs of nodes i_t that are already at their maximum capacity (i.e., those i_t for which the number of cars parked there, U_{it}, is equal to station i's maximum capacity C_i), as using such an arc in a later iteration would yield a violation of the capacity constraints. The latter modifications ensure that any path from the remaining subgraph of G can be added to a partial solution without violating any capacity constraints. The algorithm terminates if all cars have been used, if the remaining budget is insufficient for buying another car, or if no profitable path was found.

Resource-constrained shortest path algorithm. As noted above, the path-based heuristic relies on finding profitable battery-feasible paths through the location graph whose additional costs will not exceed the remaining budget. We identify such a path by solving a *resource-constrained shortest path* (RCSP) problem on location graph $G = (V, A)$ (or on a subgraph (V', A') obtained from removing certain arcs, see above) in which we associate costs $c_a \in \mathbb{Z}$ and battery consumption $b_a \in \mathbb{Z}$ to each arc $a \in A$. Initialization arcs $a \in A^{\mathrm{I}}$ have zero cost and battery consumption, i.e., $c_a = b_a = 0$. Waiting arcs $a = (i_t, j_{t'}) \in A^{\mathrm{W}}$ have cost $c_a = 0$ and battery consumption $b_a = -(t' - t)\rho$. Note that this negative consumption corresponds to an increase in charge state, i.e., recharging. Finally, trip arcs have cost and battery consumption equivalent to the negative profit and to the battery consumption of their corresponding trip, respectively. Thus, for every trip arc $a \in A_k^{\mathrm{T}}$ of any trip $k \in K$, we have $c_a = -p_k$ and $b_a = b_k$.

A dynamic-programming-based labeling algorithm is used for finding paths from the root vertex to each station's final (w.r.t. time) vertex that consume at most $B^{\max}$ battery. While such a path is guaranteed to be connected and battery-feasible, its additional costs may exceed the remaining budget since the path may use stations that are not yet opened or for which an additional charger may need to be installed. Thus, we also keep track of these aspects in our RCSP algorithm to ensure that the path it finds can be added to the preliminary solution.

In our dynamic programming procedure, we also exploit the fact that G is acyclic by considering the nodes $i_t \in V$ in order of non-decreasing time values t (which is a topological ordering). Each node i_t therefore needs to be considered only once in the algorithm when extending the labels associated to i_t by considering all outgoing arcs.

Besides these modifications, our labeling algorithm for solving the RCSP is pretty standard. Each label $\ell = (\text{cost}_\ell, \text{battery}_\ell)$ has associated costs (cost_ℓ) and charging state (battery_ℓ). For each node i_t, all dominated labels are removed whenever a new label is added. A label ℓ' is dominated by another label ℓ if $\text{cost}_\ell \leq \text{cost}_{\ell'}$ and $\text{battery}_\ell \geq \text{battery}_{\ell'}$, with at least one of the inequalities being strict.

5.3 Flow-based heuristic

A potential disadvantage of the path-based heuristic stems from the fact that it only considers one path in each iteration, which is then greedily added to the solution. Thus, it exhibits a rather local view of the instance. In contrast, the flow-based heuristic (FH) which we propose as an alternative to PH, aims to identify promising vehicle paths from a more global perspective. Its main idea is to first neglect the budget- and battery-constraints and compute an optimal solution to the resulting budget- and battery-unconstrained instance via a minimum-cost flow algorithm in the location flow network, see Section 3.1. PH is then used to obtain a solution that also respects the budget and battery constraints.

To this end, FH computes a solution to an instance I of the CSLP in the following three main steps:

1. Compute an optimal solution $(S'', \boldsymbol{C''}, H'', K'', \mathcal{H}'')$ to instance I' obtained from I by relaxing the battery and budget constraints.

2. Compute solution $(S', \boldsymbol{C'}, H', K', \mathcal{H}')$ by extracting profitable battery- and budget feasible car routes with the aforementioned RCSP algorithm, using only those trip arcs $A_{K''}$ that were used by the solution from the first step. This is equivalent to applying PH to a problem instance that contains no trip arcs except those in $A_{K''}$.

3. Apply a version of PH that starts with solution $(S', \boldsymbol{C'}, H', K', \mathcal{H}')$ (and an appropriately modified location graph where all trip arcs of already accepted trips, as well as incoming arcs of full vertices, are removed) to add further routes. This step is only performed in case sufficient budget and at least one unused car remain after the previous step.

6 Computational experiments

In our experiments, we consider five variants of the ILP models presented in Section 4: the multi-commodity flow formulation from Section 4.1.1 with continuous battery tracking (FC), battery-infeasible path cuts (FP) and the battery graph flow model (FG), as well as the connectivity cut formulation from Section 4.1.2 with continuous battery tracking (CC) and battery-infeasible path cuts (CP). We forwent including a variant that combines the connectivity cut formulation with the battery graph flow model, since the latter's large size conflicts with the former's goal of keeping the model small. The results shown in Section 6.2 suggest that the performance of such a model would not have compared favorably to the other variants anyway. Table 1 gives a concise overview of these models.

For those models where some constraints are added dynamically during branch-and-cut (i.e., FP, CC, CP), the corresponding constraint separation as described in Section 5.1 was done for every integral and fractional solution. Also, all valid inequalities (33) are added statically to the models CC and CP.

Before starting to solve the respective ILPs, we solved each instance with both the path-based heuristic (PH) described in Section 5.2 and the flow-based heuristic (FH) from Section 5.3 and provided the best of these two solutions to the ILP solver as a starting solution. We additionally used the two heuristics throughout the branch-and-bound procedure to derive integer-feasible solutions from the current fractional ones. To this end, we rounded the values of z_i variables, which correspond to the number of chargers built at each station, to the nearest integer value and used these rounded values as the corresponding station's maximum capacity in our heuristic algorithms. We then provided the best resulting solution to the ILP solver as the new incumbent solution whenever its profit exceeded the best currently known. We ensured that the heuristics were only run once for each set of rounded z_i values by saving their hashes and checking for duplicates.

All algorithms were implemented in C++. The experiments were performed on a single core of an Intel Xeon E5-2670v2 processor with 2.5 GHz that had both Hyper-Threading and Turbo Boost disabled. To solve the ILP models in our algorithms, we used IBM ILOG CPLEX 12.6.3. We limited the number of threads to one, imposed a CPU time limit of 3 h for the grid graph instances and 6 h for the Vienna instances, and limited the usable memory to 6 GiB.

Table 1: Overview of the models that were used in our computational experiments.

	battery tracking		
car movement	continuous (Section 4.2.1)	infeasible path cuts (Section 4.2.2)	battery graph (Section 4.2.3)
multi-commodity flow (Section 4.1.1)	FC	FP	FG
connectivity cut (Section 4.1.2)	CC	CP	–

Instances that reached the memory limit were included in our results as if they had reached the time limit (i.e., we considered their runtime to be 10 800 s and 21 600 s, respectively). Finally, we set the branching priorities of variables $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ to 900, 1000 and 800, respectively. Other than that, default values were used for all parameters.

6.1 Benchmark instances

We used two sets of benchmark instances to test our algorithms on: ones that were generated based on grid graphs and ones that are based on data from Vienna.

Grid graph instances. Instances of this class represent a planning period of 24 h. Their underlying street network is a two-dimensional 50×50 grid graph where each vertex is connected to its upper, lower, left and right neighbor, if such a neighbor exists, i.e., $|\mathcal{V}| = 2500$ and $|\mathcal{E}| = 4900$. The length ℓ_e of an edge $e \in \mathcal{E}$ represents the walking time (in minutes) between its two incident vertices. For horizontal edges, $\ell_e = 3$, whereas for vertical edges, $\ell_e = 2$. A subset of $|S| \in \{10, 25, 50\}$ randomly selected vertices is chosen as potential charging stations. Each such station $i \in S$ is then randomly assigned an opening cost $F_i \in \{9000, 9100, \dots, 64000\}$, a maximum capacity $C_i \in \{1, 2, \dots, 20\}$ and a per-charger cost $Q_i \in \{22000, 22100, \dots, 32000\}$. The neighborhood $\mathcal{N}_i$ of each station is the set of all vertices that are reachable on foot within five minutes (i.e., those vertices whose Manhattan distance in the grid graph to the station is at most five).

Finally, 1000 random trips are added to the instance file. The origin o_k and destination d_k of each trip $k \in K$ is randomly selected from the set of vertices within the neighborhood of at least one station, thus ensuring that every trip can at least potentially be accepted. For the other trip parameters, we distinguish between *short* and *long* trips. The battery consumption b_k (in percent of the car's battery capacity) of each trip $k \in K$ is a randomly chosen integer from the interval [5, 25] for short trips and from [25, 75] for long trips. Proportional to b_k, we set the trip's duration in minutes to $\Delta_k = \lceil \sigma_k \cdot b_k \rceil$, where $\sigma_k \in [3, 4.5]$ is a trip-specific random scaling factor. Using this duration, we chose a random start time $s_k \in \{0, 1, \dots, 1440 - \Delta_k\}$, which also fixes the end time to $e_k = s_k + \Delta_k$, and the profit to $p_k = 30 \cdot \Delta_k$ cent for each trip. The sets of potential start and end stations $N(o_k)$ and $N(d_k)$ are limited to the (at most) three closest stations within 5 min walking distance of o_k and d_k, respectively.

For each number of potential charging stations $|S| \in \{10, 25, 50\}$, we generated ten different instance files: five *long* ones with 667 long and 333 short trips and five *short* ones with 667 short and 333 long trips. To ensure that instances with fewer than 1000 trips can easily be created from these files while maintaining the 2 : 1 ratio of long to short (or short to long) trips, they are alternately added to *long* instance files in a $(long, long, short, \dots)$ sequence and to *short* ones in a $(short, short, long, \dots)$ sequence.

Vienna instances. Our real-world instances are based on data from the city of Vienna. The street network is derived from the city's OpenStreetMap data. We removed all streets that are inaccessible to pedestrians, such as motorways, since the street network is only used for finding nearby stations that can be reached on foot. Potential charging stations are then placed at supermarkets, parking garages and subway stations. Each station is randomly assigned an opening cost $F_i \in \{9000, 9001, \dots, 64000\}$, a capacity $C_i \in \{1, 2, \dots, 10\}$ and a per-charger cost $Q_i \in \{22000, 22001, \dots, 32000\}$. As for the random instances, their neighborhood is the set of vertices reachable on foot within five minutes, assuming a walking speed of 1.34 m/s.

All trip data is derived from actual taxi trips that took place over the course of one week in spring 2014. Each trips' origin, destination, start and end time is taken directly from these taxi trips, whereas the profit is set to 30 cent per minute as for the artificial instances. The battery consumption is calculated from the trip's duration and actual travel distance, based on a formula similar to the one used by Bektaş and Laporte (2011), and then rounded up to the next percentage point. As for the artificial instances, we only considered the (at most) closest three stations within walking distance of the origin and destination as potential start and end stations. Altogether, the instance contains 693 stations and 37965 trips.

6.2 Computational results on grid graph instances

From each of the instance files described above, we generated several concrete instances. To create instances of different sizes, we set the set of trips K to be equal to the first $10, 25, 50, 75, 100$ or 200 trips, respectively, given in the

Table 2: Average runtimes (avg. time) and optimality gaps (avg. gap) in percent, numbers of solved instances (#optimal) and relative qualities of the solutions found by the path heuristic (PH) and the flow heuristic (FH) compared to the best known solutions in percent for grid graph instances. Results are grouped by numbers of potential stations and trips, with 360 instances per row. Best values are marked bold.

		avg. time [s]					avg. gap [%]					#optimal					% best	
$\|S\|$	$\|K\|$	FC	FP	FG	CC	CP	FC	FP	FG	CC	CP	FC	FP	FG	CC	CP	PH	FH
10	10	**0**	**0**	**0**	**0**	**0**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**360**	**360**	**360**	**360**	**360**	**98.69**	98.00
	25	**0**	**0**	1	**0**	**0**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**360**	**360**	**360**	**360**	**360**	**91.02**	88.30
	50	**15**	61	396	392	191	**0.00**	**0.00**	0.05	0.03	0.02	**360**	**360**	356	354	356	**88.36**	85.38
	75	**1275**	2302	6482	4065	3623	**0.18**	0.47	4.59	1.19	1.20	**333**	312	182	240	251	**82.94**	81.20
	100	**5223**	6173	10055	7568	6819	**2.01**	2.83	29.44	3.81	3.69	**219**	182	40	133	144	**82.30**	80.19
	200	10479	10342	10800	10554	**10274**	**12.47**	13.68	212.43	19.91	21.64	18	19	0	13	**24**	**81.08**	79.65
25	10	**0**	**0**	**0**	**0**	**0**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**360**	**360**	**360**	**360**	**360**	96.43	**97.25**
	25	**0**	**0**	1	**0**	**0**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**360**	**360**	**360**	**360**	**360**	**94.48**	94.22
	50	**6**	18	86	44	20	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**360**	**360**	**360**	**360**	**360**	**89.49**	88.75
	75	**504**	703	2907	1376	1081	**0.06**	0.07	0.71	0.25	0.17	**352**	**352**	301	329	336	**86.39**	85.02
	100	**2445**	3317	8048	4113	4200	**0.42**	0.73	11.61	0.94	1.00	**301**	284	126	243	239	**84.06**	83.13
	200	**9367**	9650	10772	9897	9408	**4.71**	7.27	132.96	8.42	10.29	**67**	53	1	39	59	**80.93**	79.11
50	10	**0**	**0**	**0**	**0**	**0**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**360**	**360**	**360**	**360**	**360**	**98.64**	98.37
	25	**0**	1	6	**0**	**0**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**360**	**360**	**360**	**360**	**360**	**96.57**	95.34
	50	**5**	11	56	14	21	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**360**	**360**	**360**	**360**	**360**	**93.95**	92.38
	75	**279**	611	1424	800	467	**0.01**	0.04	0.14	0.06	0.02	**356**	353	329	343	354	**91.67**	90.29
	100	**1096**	1523	4150	1721	1582	**0.21**	0.27	3.20	0.36	0.33	**329**	322	255	311	317	**88.89**	87.83
	200	**7895**	8741	10747	9234	8631	**2.20**	4.34	102.32	4.86	5.55	**122**	89	8	63	91	**81.56**	81.20

instance file. We then partitioned the planning period into 24, 48 or 96 time periods (with a duration of 60 min, 30 min and 15 min, respectively) and rounded the start and end time of each trip down and up accordingly. The number of available cars has been set to $|K|/2$, $|K|/5$ or $|K|/10$. These cars have an acquisition cost of $F_c = 20000$, a battery capacity of $B^{\max} = 100$ (since each trip's battery consumption is given in percent of the car's battery capacity) and a recharge rate of $20/h$, resulting in $\rho = 20$, $\rho = 10$ or $\rho = 5$ depending on the considered partition of the planning period. For each such set of parameters, we generated a *proportional* instance where each trip has the associated profit p_k given in the instance file and a *uniform* one where each trip has a profit of $p_k = 1$.

We first solved each of these instances with algorithm FC and without a budget limit. Using the budget W' required by the resulting solution (after removing unused cars, stations and chargers during post-processing) as a baseline, we set the budget limit for our instances to $W = 0.3 \cdot W'$ or $W = 0.6 \cdot W'$.

Each of the resulting 6480 instances was then solved by each of the five algorithms considered. A summary of the obtained performance-related results is given in Table 2 and Figure 5, while Table 3 summarizes important solution characteristics.

We first observe that increasing the number of trips seems to make the problem harder to solve. Whereas all instances with only 10 or 25 trips can easily be solved by each of our algorithms, this is no longer the case for those with 50 or more trips. These observations are consistent with our expectations that the difficulty of solving an instance increases with its size. Specifically, as the number of trips increases, the number of feasible car routes can increase significantly, which in turn enlarges the search space that must be explored during branch-and-bound.

Somewhat surprisingly, however, increasing the number of potential charging stations apparently makes the problem easier to solve, even though this increases the size of all considered ILP models. We assume that this is due to the way in which we generated our grid graph instance files: since every trip's origin and destination are placed within the neighborhood of at least one station, having only few potential charging stations leads to each of them being a candidate start or end station for a large number of trips. Additionally, since the neighborhoods of these few stations are rather unlikely to overlap, trips are more likely to only have few (or one) start and end station(s). Thus, each strategic decision related to a station (such as whether to open it or not, or how many chargers to build there) can influence many trips. Additionally, since the number of cars that can be at a station at any point in time increases with trip density, it becomes more likely that a station's assigned size or maximum capacity actually becomes constraining. Altogether, this might explain why despite the model becoming smaller, it also becomes harder to solve.

Comparing the different variants of our ILP model, algorithm FC, which uses the multi-commodity flow formulation with continuous battery tracking, shows the best overall performance. For each of the instance subsets, it has the lowest average runtime, the lowest average optimality gap after the algorithm finishes (due to optimality, time or memory limit) and the highest number of instances solved to optimality. This is probably due to the way we construct the location graph (see Section 2.1), which keeps its size to a minimum, we also keep the size of the resulting ILP

Table 3: Average number of stations opened, vehicles purchased and trips accepted by the best solutions found by each of the five algorithm variants. Results are grouped by the number of potential stations $|S|$ and the number of trip requests $|K|$.

		avg. $\lvert S' \rvert$					avg. H'					avg. $\lvert K' \rvert$				
$\lvert S \rvert$	$\lvert K \rvert$	FC	FP	FG	CC	CP	FC	FP	FG	CC	CP	FC	FP	FG	CC	CP
10	10	4	4	4	4	4	2	2	2	2	2	3	3	3	3	3
	25	6	6	6	6	6	4	4	4	4	4	9	9	9	9	9
	50	8	8	8	8	8	7	7	7	7	7	21	21	21	21	21
	75	8	8	9	8	9	10	10	9	10	10	35	35	35	35	35
	100	9	9	9	9	9	13	13	12	13	13	51	51	48	51	51
	200	9	9	10	9	8	19	19	16	18	16	99	98	87	95	81
25	10	4	4	4	4	4	2	2	2	2	2	3	3	3	3	3
	25	9	9	9	9	9	5	5	5	5	5	8	8	8	8	8
	50	14	14	14	14	14	8	8	8	8	8	20	20	20	20	20
	75	17	17	17	17	17	12	12	12	12	12	31	31	31	31	31
	100	19	19	19	19	19	15	15	15	15	15	45	45	44	45	45
	200	22	22	23	22	21	27	26	21	26	24	99	97	85	96	90
50	10	5	5	5	5	5	2	2	2	2	2	3	3	3	3	3
	25	10	10	10	10	10	5	5	5	5	5	8	8	8	8	8
	50	19	19	19	19	18	10	10	10	10	10	18	18	18	18	18
	75	24	24	24	24	24	14	14	14	14	14	29	29	29	29	29
	100	29	29	29	29	29	18	18	17	18	18	40	40	40	40	40
	200	37	37	40	37	37	30	30	24	29	29	91	89	79	88	88

model manageable so that it can still be efficiently solved by CPLEX.

Algorithm FP, which instead uses battery-infeasible path cuts, also shows good performance across the board, but usually trails behind FC. We assume that this is a result of the added cuts being relatively weak (essentially only prohibiting a single infeasible solution), which reduces the strength of the LP relaxation too much to compensate for the reduced model size. This suggests that mixing two different strategies for designing ILP models, i.e., using strong compact formulations on one hand and dynamically adding cuts to a smaller initial model on the other hand, does not pay off for these instances.

Interestingly, the relative performance of these two approaches for ensuring battery feasibility seems to be somewhat switched for those algorithms using the connectivity cut formulation. Here, algorithm CP, which uses battery-infeasible path cuts, on average often outperforms algorithm CC, which is based on the continuous battery formulation. While there are some instance subsets on which CC shows the better average performance, Section 6.2 shows that CP is consistently able to solve more instances to optimality within a certain time limit than CC. This again shows that mixing strategies does not seem to work well for these instances.

However, all four algorithms discussed so far show a relatively good performance across this first set of benchmark instances. Up to $|K| = 100$, a large majority of them can be solved to optimality within the given time limit of 3 h and the average optimality gaps remain below 4% even for the more difficult sets of instances (and below 1% for most others). It is only at around $|K| = 200$ that they reach their limits as to which instances they can still somewhat reliably solve. Sections 6.2 and 6.2 also show that as the instances become harder to solve, the performance differences between the algorithms become somewhat smaller.

Algorithm FG, however, which uses the multi-commodity flow formulation and the battery graph flow formulation, shows significantly poorer performance than the aforementioned four algorithms. While its performance across the different instance sets follows the same pattern as the other four, it trails them quite significantly with respect to average runtime, average gap and number of instances solved to optimality. We assume that this is because this model is not sufficiently stronger than the others to counteract its comparatively large increase in size. Thus, since this third way of ensuring battery feasibility does not look promising, we refrain from testing a connectivity cut model with the battery graph flow formulation.

In addition to comparing the ILP models, we also analyzed the performance of the two heuristic algorithms PH and FH. Since their runtime was negligible (usually below 10 ms) for these instances, we focus our analysis on the resulting solution quality. For each subset of instances, the average solution quality of both heuristics is very similar, with the path-based heuristic PH usually finding a slightly better solution on average. Both heuristics also seem to follow the same difficulty pattern as the ILP-based algorithms, finding higher-quality solutions for those instances that are easier to solve by the exact methods (i.e., those with a lower number of trips and a higher number of stations). Their solution quality is also quite good in general, averaging at around 90% of the best known solution (which is often actually an optimal one).

As Table 3 shows, the solutions found by the five different algorithmic variants are quite similar w.r.t. the number

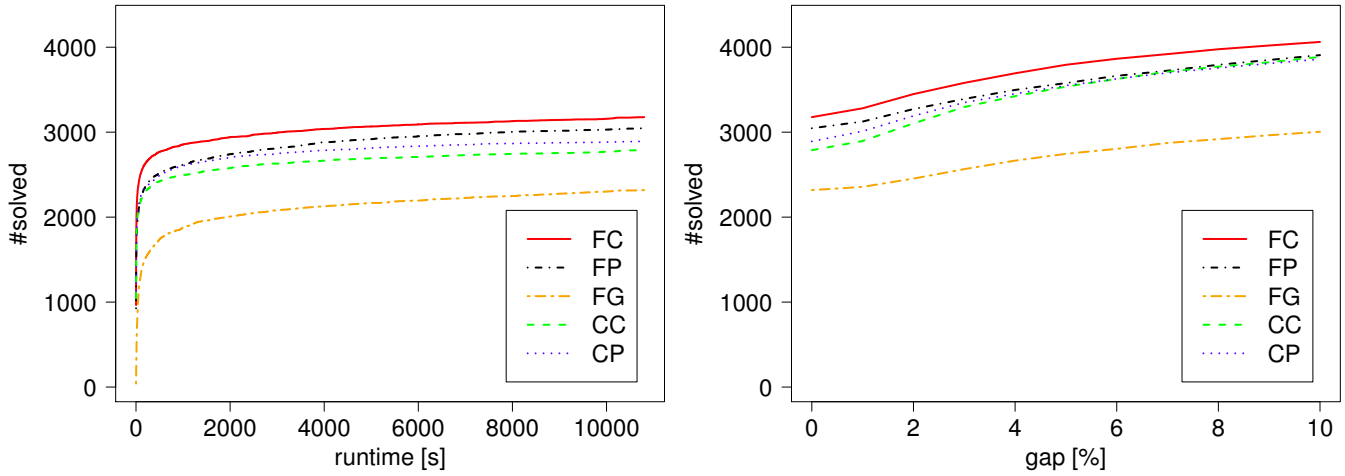


Figure 5: Cumulative runtime and gap charts for artificial instances with $|K| \geq 50$

of stations opened, vehicles purchased and trips accepted by them. This is, of course, not very surprising for those instances with $|K| \leq 75$, since these are solved optimally or with low single-digit optimality gaps by all five models. This similarity still largely holds for instances with $|K| = 100$, although the number of cars and trips in the solutions found by algorithm FG is sometimes a bit below that of the other solutions. For instances with $|K| = 200$, these differences become a bit more pronounced, with the solutions provided by algorithms FG and CP diverging most frequently and strongly. Also, the number of stations opened in FG's solutions is consistently a bit higher than in the other algorithm's solutions.

These emerging differences can likely be explained by the more frequent suboptimality of the identified solutions and their, on average, larger optimality gaps. These differences, however, do not seem to directly correlate with the computational performance of the respective algorithm. While it is mostly model FG (i.e., the one that mostly performs worst) whose solutions deviate from the rest, variant CP shows the strongest deviation for instances with $|S| = 10$ and $|K| = 200$, where it is able to solve more instances to optimality than all the other models.

Generally, we observe that as the number of trip requests $|K|$ increases, so does the relative amount of trips that are accepted (i.e., $|K'|$). For the smallest instances, this fraction is around 30%, but it steadily increases to around 50% for instances with 200 trip requests. The number of vehicles purchased also increases with $|K|$, but does so more slowly, which results in an increasing trip-to-car ratio. Likewise, the number of opened stations slowly increases with the number of available trip requests, as more and more stations are needed to cover all the origins and destinations of the trips that our algorithms accept.

Unsurprisingly, the number of opened stations $|S'|$ also increases with the number of potential stations $|S|$, as does the number of purchased vehicles. The number of accepted trips, however, actually decreases as more potential stations become available. An explanation for this behavior can be found in how these instances were constructed. Since all trips' origins and destinations were placed within walking distance of at least one station, those instances with fewer potential stations simply have a much higher density of these points within their neighborhoods. Thus, fewer are needed to cover the same number of trips.

6.2.1 Analysis of parameter impact.

We also analyzed the effects of parameters other than the number of trips and stations on the performance of our algorithms. As shown above, instances with 10 or 25 trips are trivial to solve and those with 50 trips can be usually solved quite rapidly. Thus, we exclude these instances from the following analyses, in which we focus only on the most promising algorithms FC, CC and CP, since FP does not seem to improve noticeably upon FC and since FG, as argued before, showed rather poor performance. In all following box plots that plot average gaps, we cut off the graph around the uppermost whisker to better visualize the central part of the box plot. The number of outliers in each category with a gap above the cutoff is printed above the corresponding plot in parentheses.

We first analyze whether using a uniform profit for every trip instead of one that is proportional to the trip's duration has any impact on the performance of our algorithms. Figure 6 shows that instances with uniform profit are easier to solve than those with proportional profit, despite the fact that the former may introduce additional symmetries into the solution space.

Next, we analyze whether the number of time periods into which the planning period is partitioned influences the algorithms' performance. As Figure 7 shows, increasing the number of time periods seems to make the problem easier to solve in many cases, especially for algorithm FC. This effect is somewhat less pronounced for algorithms CC

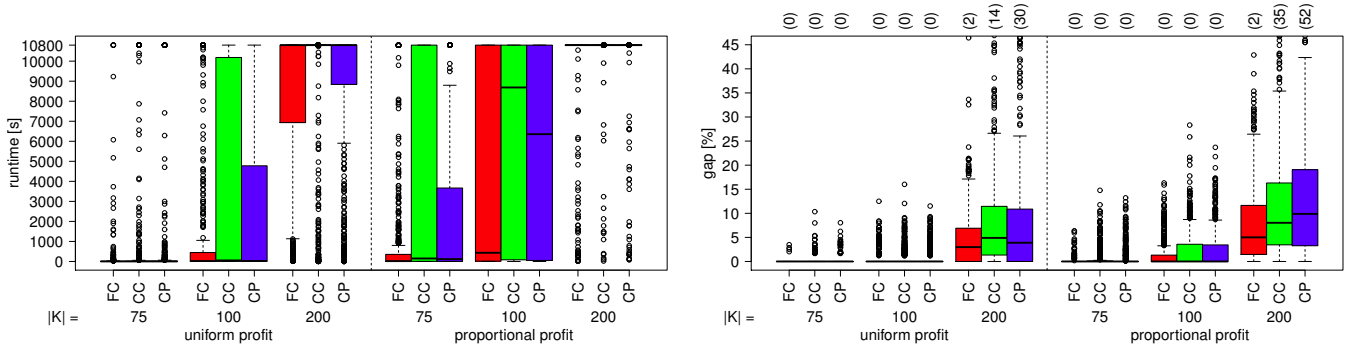


Figure 6: Distributions of runtimes [s] and optimality gaps [%] for grid graph instances with $|K| \geq 75$ grouped by profit characteristic (uniform, proportional) and $|K|$. Gap plots are cut off at 45%, with the number of outliers in each category printed above its plot in parentheses.

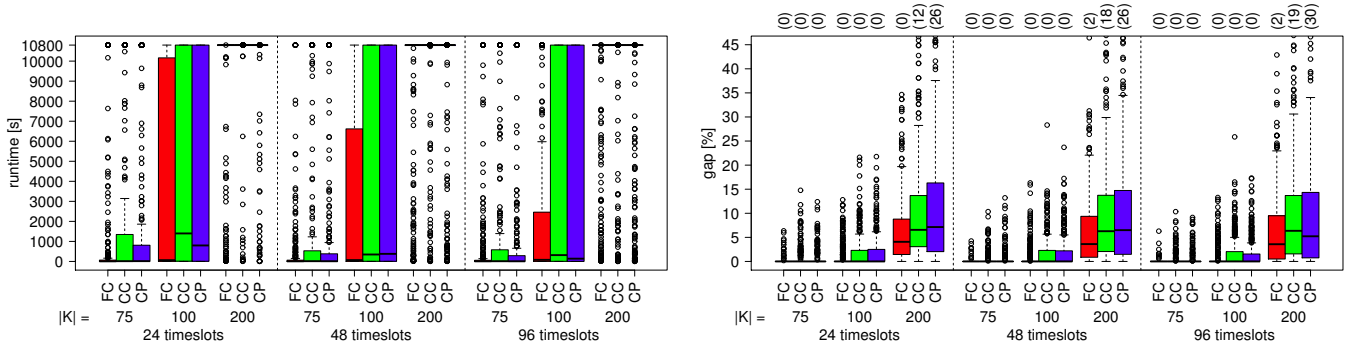


Figure 7: Distributions of runtimes [s] and optimality gaps [%] for grid graph instances with $|K| \geq 75$ grouped by the number of timeslots (1, 2 or 4 per hour) and $|K|$. Gap plots are cut off at 45%, with the number of outliers in each category printed above its plot in parentheses.

and CP, but they too seem to benefit from a more fine-grained and thus more realistic model of time. This is a bit surprising, since one would expect an increase of the size of the location graph and the number of battery variables for models FC and CC with increasing granularity of the planning period. However, as shown in Theorem 1, the location graph's size grows at most linearly with the number of trips $|K|$. This obviously also holds for the number of *relevant* points in time (those at which trips start or end) at which we need to keep track of battery state changes. One possible explanation for why the problem actually becomes easier to solve as the number of time periods increases is the fact that previously infeasible car routes can become feasible when this happens – either because trips that previously overlapped (due to rounding down and up the associated start and end times) now no longer do, or because some additional recharging between trips has made them newly battery-feasible.

As Figure 8 shows, the average battery consumption of each trip, which for our grid instances is mainly determined by whether they are *long* or *short* instances, also significantly impacts the performance of our algorithms. *Long* instances, two-thirds of whose trips are long ones with a relatively high battery consumption between 25% and 75%, are clearly harder to solve than *short* instances that instead contain this many short trips with a battery consumption between 5% and 25% each. Such a behavior is to be expected, since an increased battery consumption per trip can significantly increase the number of potential car routes that are battery-infeasible. This parameter has an especially strong impact on algorithm CP, probably because it directly influences the number of battery-infeasible path cuts that must be added to the model.

Another instance parameter that has a significant impact on our algorithms is the number of cars that are available for purchase. Figure 9 shows that as the number of available cars in a set of instances increases, so does the difficulty of solving them. Again, this is to be expected, since many variables in our ILP models are car-indexed and therefore, their number and thus the size of these models increases linearly with the maximum number of cars. Furthermore, adding more available cars introduces additional symmetries into our models, since the fleet is homogeneous.

The final parameter we consider in our analysis is the available budget. Figure 10 shows that doubling the budget increases the runtime that is required for solving instances for all three algorithms. Optimality gaps, on the other hand, actually become smaller as the available budget increases. These conflicting behaviors of the two performance indicators can perhaps be explained by how the budget influences the feasibility of candidate solutions. Increasing the available budget enlarges the set of feasible strategic decisions (i.e., which stations to open at which size, how many

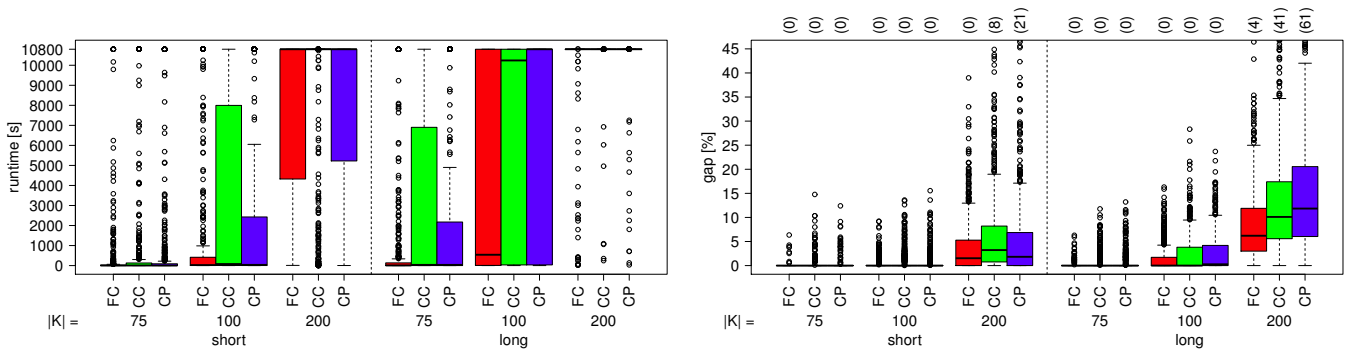


Figure 8: Distributions of runtimes [s] and optimality gaps [%] for grid graph instances with $|K| \geq 75$ grouped by majority trip length (short, long) and $|K|$. Gap plots are cut off at 45%, with the number of outliers in each category printed above its plot in parentheses.

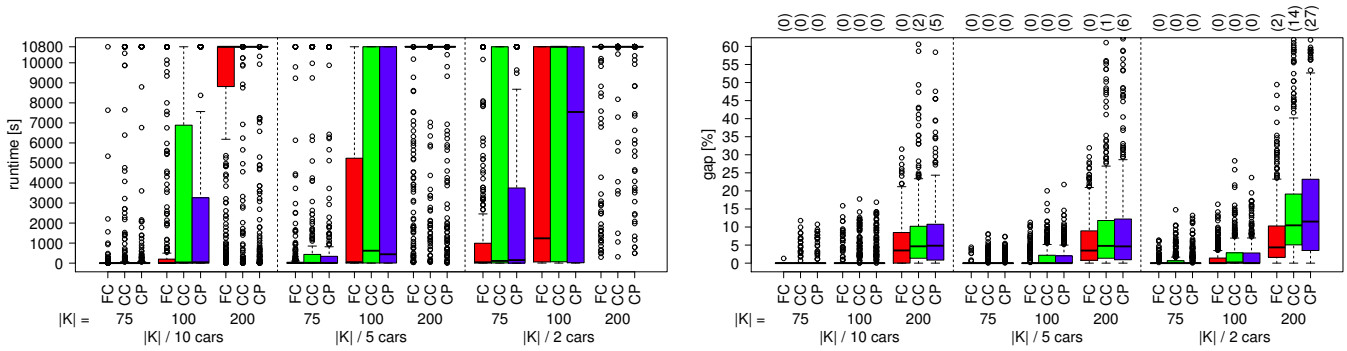


Figure 9: Distributions of runtimes [s] and optimality gaps [%] for grid graph instances with $|K| \geq 75$ grouped by number of cars (one car per 2, 5, or 10 trips) and $|K|$. Gap plots are cut off at 60%, with the number of outliers in each category printed above its plot in parentheses.

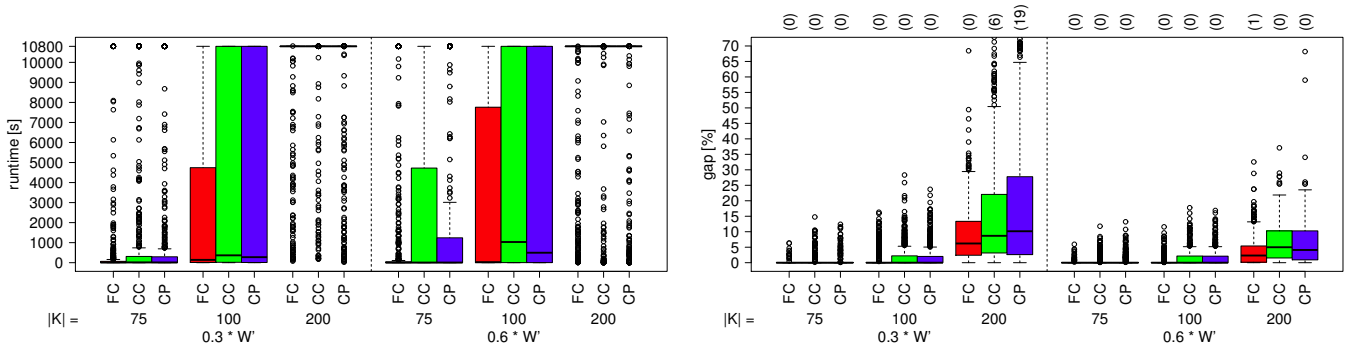


Figure 10: Distributions of runtimes [s] and optimality gaps [%] for grid graph instances with $|K| \geq 75$ grouped by budget ($0.3 \cdot W'$ or $0.6 \cdot W'$) and $|K|$. Gap plots are cut off at 70%, with the number of outliers in each category printed above its plot in parentheses.

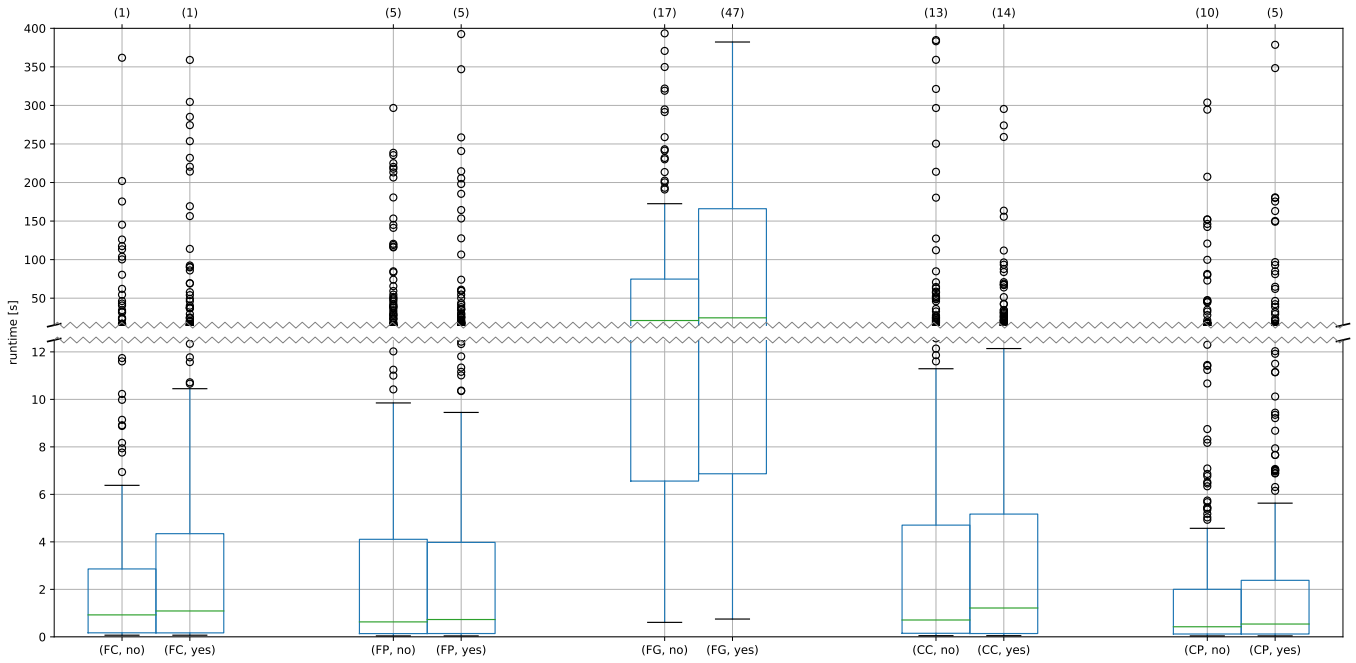


Figure 11: Boxplots showing the runtimes (in seconds) for the five different ILP model variants (see Table 1) with ("yes") and without ("no") symmetry breaking constraints (43) for grid graph instances with $K = 50$ and $T = 96$. The number of outliers above 400 s is given above each boxplot in parentheses.

cars to buy), which can make it easier to find good solutions (resulting in smaller gaps). However, it also increases the size of the search space that must be explored by branch-and-bound, which in turn increases the time required for doing so.

6.2.2 Symmetry breaking

Since we assume the vehicle fleet in our car sharing system to be homogeneous, the models we presented in Section 4 all exhibit some symmetries regarding their vehicle-indexed variables. Specifically, for every solution with H' vehicles, there exist $\binom{H}{H'} \cdot H'!$ equivalent variable assignments that only differ in which of the vehicles $\{1, \dots, H\}$ are purchased and how the solution's car routes are assigned to them.

In an effort to reduce the number of symmetric solutions, we introduced constraints

$$a_h \geq a_{h+1} \qquad \forall h \in \{1, \dots, H-1\} \tag{43}$$

to ensure that in every solution, the first H' vehicles are purchased.

We then conducted additional experiments with IBM ILOG CPLEX 12.8.0 where we compared the runtime performance of our models both with and without these additional symmetry breaking constraints, using the same 3h time limit as for our previous experiments. As Figure 11 shows, the computational performance of our models was mostly diminished by their inclusion. We assume that this is due to CPLEX already being well-versed in the handling of such symmetries in the models it is solving. Based on these results, we decided to conduct all other experiments without constrains (43).

6.3 Computational results on Vienna instances

Due to the large number of potential charging stations and trips contained in the full instance, we restricted ourselves to three smaller areas of operation: area *D5* covering districts 1, 6, 7, 8 and 9, area *D6* which also includes district 5, and area *D7* that additionally includes district 4. As all these districts are relatively densely populated, these areas could serve as a convenient testing ground for an electric car sharing system – an idea that is further supported by the fact that these districts form the center of both car2go's and DriveNow's area of operation, who are the current main car sharing providers in Vienna. The number of stations and trips in the resulting instances can be further reduced by only including those stations with at least one trip potentially starting or ending within walking distance of it, as well as only trips that have at least one possible start and end station (as before, we assume a maximum walking distance of 5 min and only consider the three closest stations for determining a trip's potential start and end stations) Table 4 summarizes this data for each area of operation (AoO).

The planning period of one week is partitioned into 168, 336 and 672 time periods of length 60 min, 30 min and 15 min, respectively. As the number of available cars, we used 10, 25 and 50. These cars, whose data is based on

Table 4: Overview of the three operational areas, as well as the number of stations and trips they contain before and after preprocessing, that were considered in our experiments.

	districts							original		preprocessed	
area of operation (AoO)	1	4	5	6	7	8	9	$\|S\|$	$\|K\|$	$\|S\|$	$\|K\|$
D5	✓	✗	✗	✓	✓	✓	✓	110	144	101	108
D6	✓	✗	✓	✓	✓	✓	✓	136	257	129	209
D7	✓	✓	✓	✓	✓	✓	✓	159	589	153	480

the *Smart ED* car, have an acquisition cost of $F_c = 20000$, a battery capacity of $B^{\max} = 100$ (each trip's battery consumption is rounded up to the next full percentage point) and a recharge rate of $^{100}/_{h}$, resulting in $\rho = 100$, $\rho = 50$ or $\rho = 25$ depending on the number of considered time periods. We again considered both an instance with the trips' profits as described in Section 6.1 and one where each trip has a uniform profit of one. Finally, we used budget values of 1, 2, 3, 4 and 5 million. In total, 270 different instances of this class were included in our analysis, where we compared formulations FC, CC and CP. As for the grid graph instances, we also analyzed the impact of various instance parameters. An optimal solution for an instance with operational area D7, a budget limit of five million and ten available cars is given in Figure 12 while our findings are summarized in Tables 5 to 8.

We first note that, in general, our observations from Section 6.2 also hold for this instance set. As before, the number of trips $|K|$, which for these instances is determined by the area of operation, seems to be the principal factor in determining the difficulty of solving an instance. On a majority of instance subsets, model FC shows the best overall performance with respect to average runtimes, gaps and the total number of instances solved to optimality. However, it is still repeatedly outclassed by model CP, which shows especially good performance on those instances with the smallest area of operation, D5. Model CC is most often the worst-performing of the three, but in a few cases still comes in second or even first.

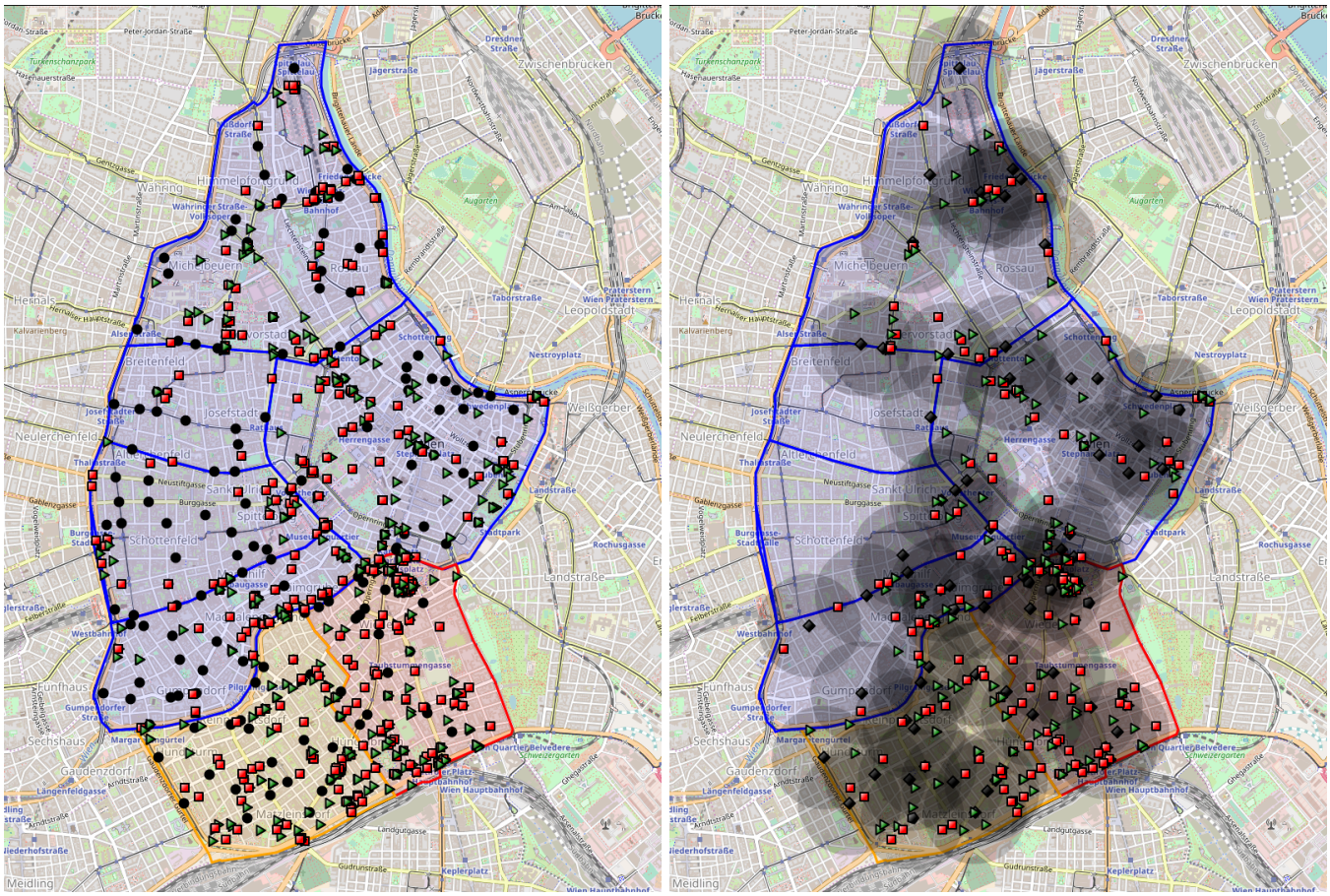
For instances with the largest area of operation D7, none of the considered algorithms is able to consistently provide high-quality solutions within the given 6h time limit. Only twelve such instances are ever solved to optimality (all by algorithm FC), all of which have the lowest number of available cars (i.e., $H = 10$) and one of the two highest budget limits (i.e., 4M or 5M). Average optimality gaps mostly exceed 50%, sometimes even going beyond 100%, except for instances with low vehicle counts and high budget limits. Thus, new algorithms would have to be developed to successfully tackle instances of that size.

When comparing the structure of the best solution found by each algorithm, we found the average number of stations opened in each of them to be very similar. Also, models CC and CP tended to, on average, purchase similar numbers of vehicles and accept similar numbers of trips in their solutions. The best solution found by model FC, however, frequently differed quite significantly from those two with respect to these two parameters, especially for the larger instances with areas of operation D6 and D7 – specifically, it often included many more purchased cars and accepted trips. This behavior is consistent with what we observed for the grid-graph-based instances, where these parameters also started to diverge as instances became larger and thus harder to solve. We assume that this simply results from the increasing suboptimality of the identified solutions that increases their dissimilarity. As the results for both the grid-graph instances and for AoO D5 show, the different algorithms mostly remain in agreement w.r.t. these parameters as long as the instances can still be solved optimally or with low optimality gaps.

Across the whole Vienna instance set, both the path- and the flow-based heuristic seem to be unable to find solutions of as high quality as for the grid instances. Compared to each other, however, the average quality of their solutions relative to the best known are very similar, although PH still appears to be the better choice in most cases. While it may seem that the heuristic works rather well on the largest instances D7, this is likely due to the fact that our algorithms are hardly ever able to solve them to proven optimality or provide good gaps – resulting in the best known solution being of rather low quality (or, in fact, the heuristic solution itself).

Table 5 shows that, as for the grid graph instances, increasing the number of time periods does not make the problem noticeably harder to solve. We refer the reader to the discussion of Figure 7 for an argument on why this is the case. Our data also shows that model CP exhibits the best overall performance on the small D5 instances, while model FC is best at solving the larger ones.

Somewhat in contrast to our previous observations, however, the data shown in Table 6 suggests that for this instance set, those instances with profit proportional to their duration are easier to solve than those with uniform profit. We assume that this difference arises from the different distributions of trip durations (and thus profits) in the two instance sets: in the grid graph instances, the actual trip durations are spread more sparsely on a much wider interval, which can lead to individual profits being much more profitable than others. Trip durations and profits for the Vienna instances, on the other hand, are much less varied, which leads to a more uniform trip landscape. However, since the profits are not all the same, the resulting ILP models have fewer inherent symmetries regarding trip selection, which may explain why these proportional instances are even easier to solve than actually uniform ones. We also observed that the number of trips accepted in the best known solution of uniform instances tends to exceed the corresponding number of proportional instances. This comes at little surprise, since for the latter, our models can



(a) Instance (b) Solution

Figure 12: Visualization of one Vienna instance (with operational area D7, $W = 5$M, $H = 10$, $T_{\max} = 672$ and proportional profit) and its optimal solution. Figure 12a shows the location of all potential charging stations (●), as well as the origin (▶) and destination (■) of each trip request. It also highlights the boundaries of the three different operational areas D5 (in blue), D6 (in blue and orange) and D7 (in blue, orange and red). Figure 12b shows the locations and sizes (◆ for one charger, ⬟ for two) of all opened stations and the origins and destinations of all accepted trips. It also visualizes circular approximations of the neighborhoods covered by each opened station as semi-translucent circles. Note that the actual walking distance on the street network was used during our experiments.

Table 5: Average runtimes and optimality gaps, numbers of instances solved to optimality, average numbers of stations, cars and trips in the best known solutions, and the qualities of both heuristic solutions relative to them for all Vienna instances. Results are grouped by the number of time periods $T_{\max}$ and the area of operation (AoO), with 30 instances per row.

		avg. time [s]			avg. gap [%]			#optimal			avg. $\lvert S' \rvert$			avg. H'			avg. $\lvert K' \rvert$			% best	
$T_{\max}$	AoO	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	PH	FH
168	D5	7287	8657	**3645**	0.85	5.73	**0.53**	24	19	**27**	36	36	35	19	17	19	69	65	68	**78.01**	73.35
	D6	**14610**	16165	14673	**7.30**	35.67	28.55	10	10	**12**	41	43	42	19	11	12	108	85	88	**67.43**	63.25
	D7	**19121**	21600	21600	**68.45**	80.62	80.45	**4**	0	0	46	46	46	11	7	7	185	146	146	**77.37**	63.17
336	D5	6481	7900	**3747**	**0.75**	6.21	0.90	23	21	**27**	36	36	35	19	17	18	69	65	68	**78.25**	73.72
	D6	**13776**	18667	14962	**7.64**	35.00	27.88	**13**	5	10	41	43	43	19	12	12	108	86	88	**67.94**	65.54
	D7	**18823**	21600	21600	111.04	81.26	**81.21**	**4**	0	0	47	46	46	10	7	7	182	148	148	**81.30**	66.99
672	D5	7184	8370	**4654**	**0.63**	6.04	2.26	24	21	**26**	36	36	36	19	17	18	69	65	67	**79.80**	74.59
	D6	**14133**	16886	16748	**8.03**	27.31	25.74	**12**	9	9	41	44	43	19	12	13	107	89	90	**66.15**	64.74
	D7	**18823**	21600	21600	106.46	89.08	**88.89**	**4**	0	0	47	47	47	10	7	7	187	152	152	**78.57**	65.08

Table 6: Average runtimes and optimality gaps, the number of instances solved to optimality, the average number of stations, cars and trips in the best known solution, and the quality of both heuristic solutions relative to it for all Vienna instances. Results are grouped by whether trips have uniform or proportional profit, and the area of operation (AoO), with 45 instances per row.

		avg. time [s]			avg. gap [%]			#optimal			avg. $\|S'\|$			avg. H'			avg. $\|K'\|$			% best	
	AoO	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	PH	FH
unif.	D5	6525	9285	**4334**	**1.01**	8.93	1.80	36	29	**40**	36	37	36	19	16	18	71	65	69	**76.80**	71.50
	D6	**13279**	19957	18710	**9.59**	41.75	36.33	**19**	5	9	41	44	43	19	11	12	112	87	89	**63.83**	61.08
	D7	**19057**	21600	21600	114.23	100.38	**100.16**	**6**	0	0	46	46	46	10	7	7	191	153	153	**77.24**	63.65
prop.	D5	7443	7333	**3697**	**0.48**	3.06	0.66	35	32	**40**	36	36	35	19	17	18	67	64	66	**80.58**	76.27
	D6	15067	14523	**12212**	**5.71**	23.56	18.45	16	19	**22**	41	42	42	19	12	13	103	87	89	**70.52**	67.94
	D7	**18788**	21600	21600	76.40	66.93	**66.88**	**6**	0	0	47	47	47	11	7	7	178	144	144	**80.92**	66.51

Table 7: Average runtimes and optimality gaps, the number of instances solved to optimality, the average number of stations, cars and trips in the best known solution, and the quality of both heuristic solutions relative to it for all Vienna instances. Results are grouped by budget W and the area of operation (AoO), with 18 instances per row.

		avg. time [s]			avg. gap [%]			#optimal			avg. $\|S'\|$			avg. H'			avg. $\|K'\|$			% best	
W	AoO	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	PH	FH
1M	D5	11218	940	**247**	2.14	**0.00**	**0.00**	12	**18**	**18**	16	16	16	8	8	8	37	37	37	**49.94**	45.81
	D6	20449	15440	**12574**	18.45	28.01	**7.37**	3	8	**9**	17	17	17	8	8	8	48	45	49	35.68	**42.59**
	D7	**21600**	**21600**	**21600**	244.98	170.00	**169.41**	**0**	**0**	**0**	17	17	17	5	2	2	84	45	45	**62.54**	46.32
2M	D5	10072	12883	**3656**	0.95	14.98	**0.00**	12	9	**18**	29	29	29	14	12	14	64	57	64	**68.41**	61.54
	D6	**17318**	18952	19358	**11.46**	57.00	58.13	**5**	3	2	31	32	32	14	7	7	90	62	61	**58.18**	52.23
	D7	**21600**	**21600**	**21600**	**108.53**	119.72	119.62	**0**	**0**	**0**	33	33	33	7	3	3	144	89	89	**67.53**	55.88
3M	D5	6328	8825	**6116**	**0.03**	8.88	2.89	**17**	12	15	38	40	38	21	17	20	77	70	75	**84.72**	79.14
	D6	**14419**	18917	17579	**5.70**	49.51	45.44	**6**	3	5	43	47	46	20	9	10	119	81	85	**67.44**	63.89
	D7	**21600**	**21600**	**21600**	**64.37**	68.20	68.21	**0**	**0**	**0**	49	49	49	10	6	6	194	152	152	**80.37**	63.14
4M	D5	**7219**	8922	7523	**0.61**	5.03	3.23	**12**	11	**12**	46	46	45	25	21	23	83	77	79	**93.63**	88.44
	D6	**11300**	18008	15450	**2.00**	20.58	18.38	**9**	3	6	53	57	57	23	15	15	136	113	116	**82.86**	77.25
	D7	**14936**	21600	21600	**35.95**	39.09	39.09	**6**	0	0	62	62	62	13	10	10	234	206	206	**89.87**	75.98
5M	D5	**83**	9973	2535	**0.00**	1.08	0.03	**18**	11	17	52	50	48	27	26	27	84	82	84	**96.75**	94.49
	D6	**7379**	14882	12344	**0.66**	8.19	7.61	**12**	7	9	60	63	61	28	21	21	145	132	133	**91.72**	86.58
	D7	**14876**	21600	21600	22.74	**21.25**	21.26	**6**	0	0	72	72	72	18	15	15	267	251	251	**95.10**	84.08

prefer to accept fewer, but more individually profitable trips, whereas they simply try to maximize the number of accepted trips for the former ones.

Table 7 shows the effect of different budget restrictions on our algorithms and the solutions they produce. It seems that increasing W up to a certain point, which appears to be around two million for these instances, makes the instances harder to solve. However, increasing it beyond that level reverses this trend and makes the instances easier to solve once again. As argued before, we assume that this is due to the budget's effects on all strategic decisions. When only very little money is available, only a few stations can be opened until the budget limit is reached, which possibly reduces the depth of the related search tree that needs to be explored. Similarly, if the available budget is sufficient to open a large number of stations, our models can simply open most of them with many chargers. When the budget is between these two extremes, however, a large number of strategic choices may need to be evaluated to determine which of them are good. Interestingly, model FC seems to work rather poorly for instances with low budget, being consistently outperformed by model CP. We also note that, somewhat obviously, the number of opened stations, purchased cars and accepted trips keeps increasing as more budget becomes available. Our two heuristics also seem to perform noticeably worse on instances with small budget values. It might be possible to remedy these effects somewhat by using the remaining budget as a separate resource in our RCSP algorithm (which we use for both heuristics) instead of simply cutting off paths that exceed it.

Finally, Table 8 shows that as for the grid graph instances, the problem becomes harder to solve as more cars are available for purchase. We again refer the reader to the previous section for a discussion on how this behavior can be explained. Interestingly, we also observed that the share of cars that are actually used in the resulting solution decreases as more cars become available. This suggests that other factors, like the available budget W or station capacities C_i, play a much more important role in limiting the number of trips that can finally be accepted. A more in-depth analysis of how these instance parameters influence the performance and resulting solution of our algorithms could be performed with multi-objective variants of our algorithms.

Table 8: Average runtimes and optimality gaps, the number of instances solved to optimality, the average number of stations, cars and trips in the best known solution, and the quality of both heuristic solutions relative to it for all Vienna instances. Results are grouped by maximum number of cars H and the area of operation (AoO), with 30 instances per row.

		avg. time [s]			avg. gap [%]			#optimal			avg. $\|S'\|$			avg. H'			avg. $\|K'\|$			% best	
H	AoO	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	FC	CC	CP	PH	FH
	D5	231	70	**41**	**0.00**	**0.00**	**0.00**	**30**	**30**	**30**	33	32	32	10	10	10	52	52	52	**83.06**	82.00
10	D6	**5407**	10834	7382	**0.43**	5.20	3.55	**26**	18	23	39	40	39	10	9	10	86	82	83	72.65	**73.16**
	D7	**13567**	21600	21600	**5.85**	66.17	65.47	**12**	0	0	45	46	46	10	6	6	192	140	140	**69.94**	59.89
	D5	6367	7181	**1255**	0.02	2.87	**0.00**	29	25	**30**	38	38	37	20	20	20	74	72	74	**77.30**	72.25
25	D6	**15512**	19524	17401	**7.07**	34.87	32.38	**9**	5	8	42	44	44	20	13	14	117	91	94	**64.99**	63.04
	D7	**21600**	**21600**	**21600**	**43.12**	90.35	90.13	**0**	**0**	**0**	47	47	47	14	8	8	200	153	153	**73.46**	60.69
	D5	14354	17676	**10750**	**2.21**	15.12	3.69	12	6	**20**	37	39	37	27	21	25	80	69	77	**75.70**	67.41
50	D6	21600	**21361**	21600	**15.46**	57.90	46.23	0	**1**	0	41	45	45	26	12	13	120	87	89	**63.89**	57.32
	D7	**21600**	**21600**	**21600**	236.98	**94.44**	94.96	**0**	**0**	**0**	47	47	47	9	8	8	162	153	153	**93.85**	74.65

7 The gain of integration

As noted in the introduction, the models and algorithms we described in Sections 4 and 5 integrate decision-making at various levels into a single optimization framework. This, of course, increases the size and complexity of the underlying ILP models and makes them harder to solve. To analyze the benefit of our integrated solution approach, we compare it to a sequential algorithm where the strategic decisions are optimized first and then used as an input to a lower-level optimization procedure that optimizes tactical and operational decisions. More precisely, the developed sequential algorithm, which is fully described in Appendix B,

1. solves a variant of the maximal covering location problem in order to decide which stations to open,
2. opens all stations selected by facility location model at maximum capacity, and then
3. optimizes all remaining decisions by applying algorithm FC (multi-commodity vehicle flow with continuous battery tracking) with fixed decisions concerning open stations and their sizes.

In addition to studying the gain of integrations, this section also analyzes the impact of changes in the anticipated user demand via a cross-validation. To enable the latter, we use a second set of instances derived from real-world data from the city of Vienna. Recall that for our case study of the city of Vienna in the previous section, we used taxi trip data from one week as a representative of the expected customer demand. In the following experiments, which we conducted with IBM ILOG CPLEX 12.8.0, the trip request forecasts are now derived from taxi trip data from four different weeks over the course of a year and include more trips. All experiments are based on instances using districts 1, 6, 7, 8 and 9 as operational area (i.e., D5), a time granularity of 15 minutes per time period, a vehicle limit of 10 cars, trip profits proportional to their duration, and budget limits of 1M, 2M and 5M. For each of the four weeks, we then sampled five different demand scenarios from their respective trip request sets, resulting in a total of 60 different instances. These instances each contain 110 potential locations for charging stations and between around 180 and 360 trip requests.

Throughout this and the next section, we will refer to the integrated and sequential algorithms as INT and SEQ, respectively. For the former, algorithm FC has been used. We also use INT^* and SEQ^* to refer to the optimal (or best known) solution found by these algorithms.

7.1 Comparison between the integrated and sequential approach

Our first set of results comparing the integrated and the sequential approach are summarized in Figure 13, which shows cumulative numbers of instances in which at least a certain profit improvement over the sequential algorithm is achieved by the integrated algorithm.

These results show that the profit obtainable with the integrated approach can increase by more than 90% when compared to the sequential one. In one half of all instances we considered, the additional profit through integrated optimization is at least 40% of the profit obtainable in the sequential algorithm's best-known solution SEQ^*, and even in the worst case, it exceeds 10%. These results clearly indicate that the additional effort of developing and solving a more complex, integrated model can yield significant improvements concerning the final solution quality for the considered problem.

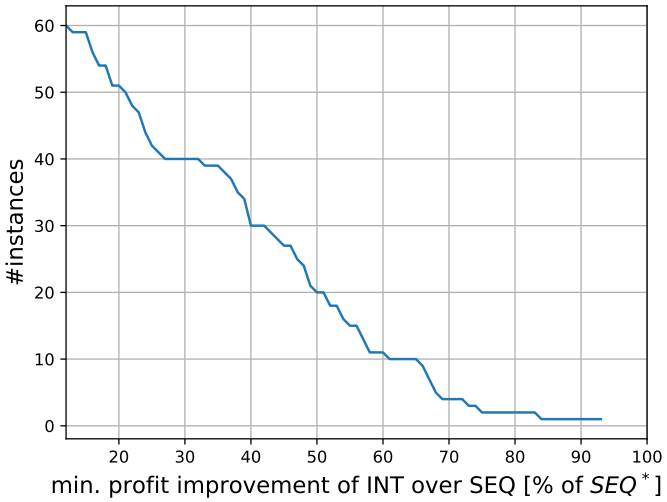


Figure 13: Reverse cumulative sum plot, showing the number of instances for which a profit improvement of at least X% could be obtained by the integrated algorithm compared to the sequential one.

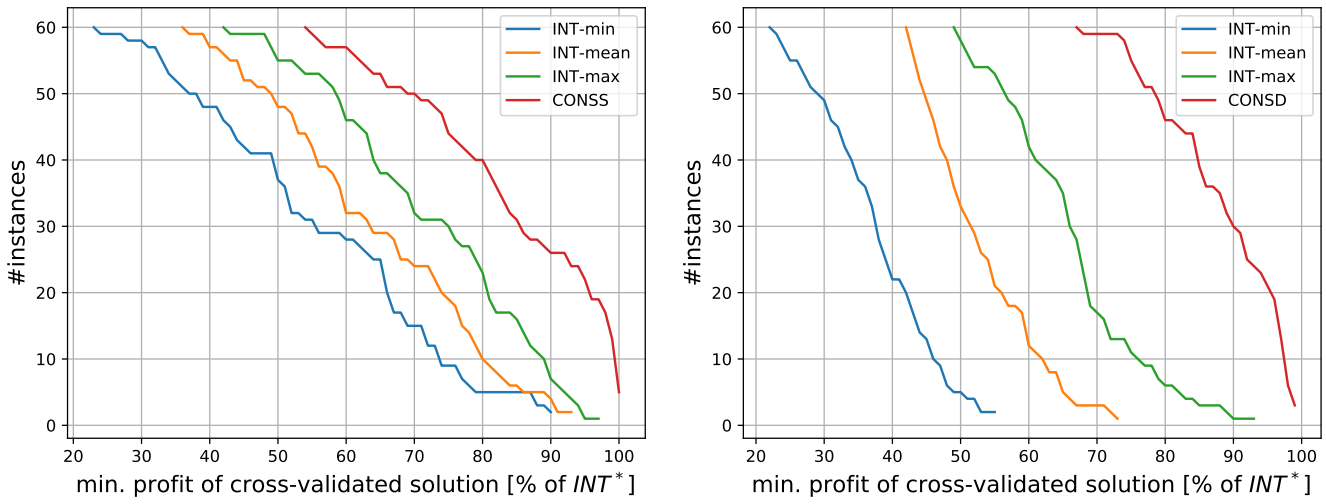
7.2 Cross-validation study

All algorithms so-far considered in this article assume to be given a deterministic demand forecast. Since this assumption may not hold in practice (i.e., the real demand may differ from the expected one), this section analyzes whether the obtained solutions (which may suffer from overfitting) are still reasonable in this case. To this end, we perform a cross-validation study in which solutions computed for a given demand forecast are evaluated on alternative demand scenarios. More precisely, we compute the maximum achievable profit for particular demands when all strategic decisions (i.e., stations and their sizes) are fixed according to an optimal solution with respect to a different scenario.

Ideally, different trip scenarios would be addressed via an appropriate extension of our formulations to a two-stage stochastic program. However, the high computational effort required for solving the deterministic problem variant (see, Section 6) makes in unlikely that a corresponding solution method would be able to solve realistic instances. As some sort of compromise between only solving single-scenario instances and a full two-stage stochastic approach, we propose an algorithm that combines solutions for individual instances into a single *consensus* solution. As will be shown below, this approach, which does not incur the entire runtime penalty of a two-stage stochastic model, works reasonably well on our instances. This consensus solution is computed by solving a restricted two-stage stochastic model where the minimum and maximum values of all first-stage decisions pertaining to stations and their sizes are set to the smallest and largest values of the corresponding variables in any of the solutions to the single-scenario instances that have been computed in a first step. In addition, only those trips accepted in a single-scenario instance are considered in this second phase, see Appendix C for more details.

In our experiments, we considered two different kinds of cross-validation. First, for each week and budget value, we evaluated the solution for each demand scenario on each of the other four demand scenarios (e.g., evaluating solution INT^* for scenario 1 on scenarios 2–5), resulting in a total of 240 cross-scenario evaluation runs. Second, for each week, budget value and scenario, we created seven individual day-long subinstances (one for each day of the week) and evaluated their INT^* solution on the corresponding full week-long instance, resulting in a total of 420 cross-day evaluation runs. Along the same lines, we created two different consensus solutions for each of the 60 instances. The first method, which we refer to as CONSS, combines the four INT^* solutions to the instance's other demand scenarios, while the second method (called CONSD) combines the seven INT^* solutions to the instance's day-long subinstances. We refer to the optimal (or, as before, best-known) solutions found by these two methods as $CONSS^*$ and $CONSD^*$, respectively.

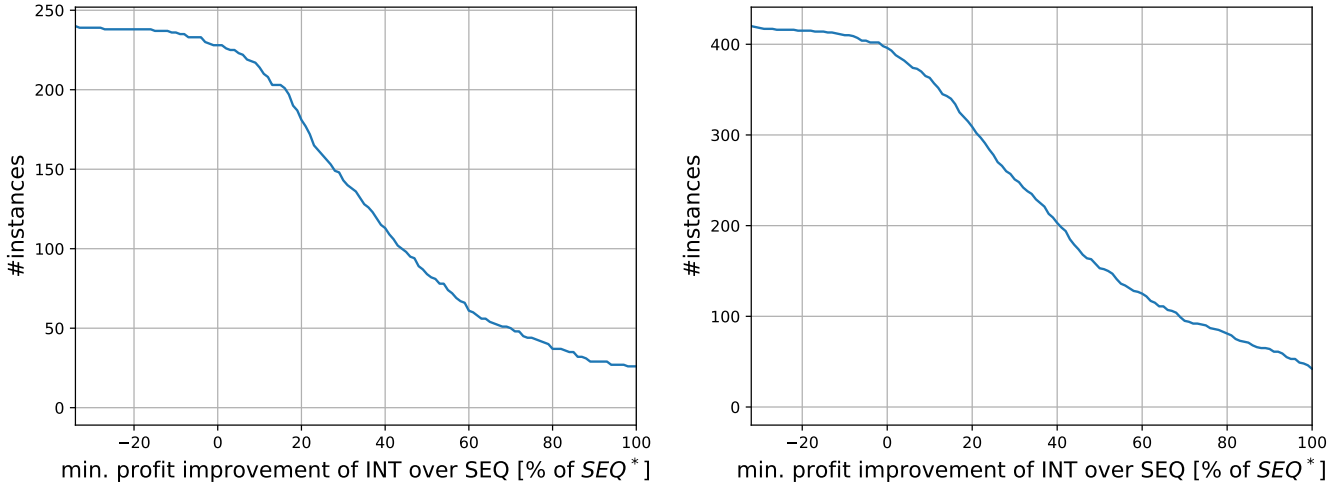
Results concerning the cross-scenario evaluation are summarized in Figure 14a and Figure 15a. We observe that the cross-validated profit (i.e., those achieved after changing the demand scenario) on average (INT-mean) lies between 40% and 90% of the profit obtained when the demand is known beforehand. Observed best (INT-max) and worst (INT-min) cases, lie approximately between 20% and 95% of the latter profit. Recall that the optimal value in the expected case, which would typically be below 100%, are not known since the required two-stage stochastic program could not be solved. Nevertheless, these results indicate that the solutions obtained by algorithm INT are reasonable, but could be further improved by assuming that the demand forecast is subject to uncertainty and incorporating this aspect into the optimization procedure. Indeed, a significant and consistent improvement is achieved by the corresponding consensus solution. Even in the worst case, more than 55% of that instance's best-known profit can still be achieved, while in a few instances, the consensus solution can even match INT^*'s profit. Overall, this approach seems to offer a good compromise between solution quality and required computational effort. Figure 15a shows that our integrated



(a) cross-scenario evaluation

(b) cross-day evaluation

Figure 14: Reverse cumulative sum plot, showing the number of instances for which the worst, average and best of the cross-validated solutions, as well as corresponding cross-validated consensus solution, achieve at least X% of the profit achieved by the best-known solution to that instance.



(a) cross-scenario evaluation

(b) cross-day evaluation

Figure 15: Reverse cumulative sum plot, showing the number of instances for which the integrated algorithm achieves at least an X% profit improvement over the sequential algorithm.

approach still mostly outperforms the sequential algorithm w.r.t. the achieved profit when cross-validation (w.r.t. different demand scenarios) is concerned. While there are a few instances where one of the cross-validated solutions from the sequential algorithm outperforms the corresponding integrated one, the latter mostly matches or outperforms the former. In one half of the instances, this gain of integration is around 40% or better.

Results from the evaluation of individual day-long subinstances on their corresponding week-long instance largely confirm these findings, as Figures 14b and 15b show. Interestingly, however, the individual plot lines are significantly steeper, i.e., the profit achievable by the worst, average, and best cross-validated solution varies a lot less across all instances. This might result from the fact that we now evaluated seven instead of four solutions on each instance, which *increases* the achievable profit range of each instance, thereby *constricting* the range that the profits achievable by the worst and best cross-validated solution will take across all instances. Again, the corresponding consensus solutions $CONSD^*$ show a much better cross-validation performance than the individual day-long INT^* solutions, achieving profits between around 70% and 100% of the corresponding week-long instance's INT^* profit.

8 Simulation study

The models and algorithms described in the previous sections all allow the car sharing operator significant freedom in the trip selection process. Specifically, the operator can choose which trip requests are accepted and, if they are,

at which stations they start and end. In practice, such a system would be implemented by means of a mobile app where the user enters the start and end address of their trip request. The system would then tell the user whether their request has been accepted and, if it has, where to pick up and drop off the rented car. Thus, we assume that the customer is indifferent to which pick-up and drop-off station is selected, as long as they are still within the allowed walking distance, which allows the operator to select these stations for them.

Alternatively, in order to provide their customers with more flexibility, the operator could offer its users a live view of all available vehicles and chargers and allow them to choose for themselves whether they want to undertake a trip, as well as where they want to start and end it. In this section, we analyze how well the solutions found by our algorithms (specifically, their strategic decisions regarding the placement and sizing of charging stations) would perform in such an alternative, *user-driven* system. To evaluate a solution's quality (i.e., the profit that can be obtained once the strategic decisions are fixed) in such a user-driven system, we developed a simulation algorithm that accepts trip requests greedily on a first-come-first-served basis. Throughout the simulation run, we assume that customers always act in their own best interest. Specifically, we assume that

1. if both a car with sufficient battery and a free charger are available within walking distance of the trip's origin and destination, respectively, a customer will always choose to perform their requested trip,
2. a customer will always start their trip at their most-preferred station that has a car with sufficient battery to perform the trip, and
3. a customer will always return the car to their most-preferred station that has a charger available.

In our experiments, we used the stations' proximity to the customer's origin and destination as a preference measure, with customers preferring closer stations to farther-away ones. A detailed description of this greedy simulation procedure can be found in Appendix D. Using the previously described simulation algorithm and the instances described in Section 7, we evaluated the solutions found by the sequential and the integrated algorithm, as well as the two consensus solutions (as described in Section 7), and compared their quality (i.e., achievable profit) to the optimal (or best-known) solution INT^* found by the regular integrated model. Throughout this section, we will refer to these four different greedily evaluated solutions as $SEQ^\dagger$, $INT^\dagger$, $CONSS^\dagger$ and $CONSD^\dagger$, respectively.

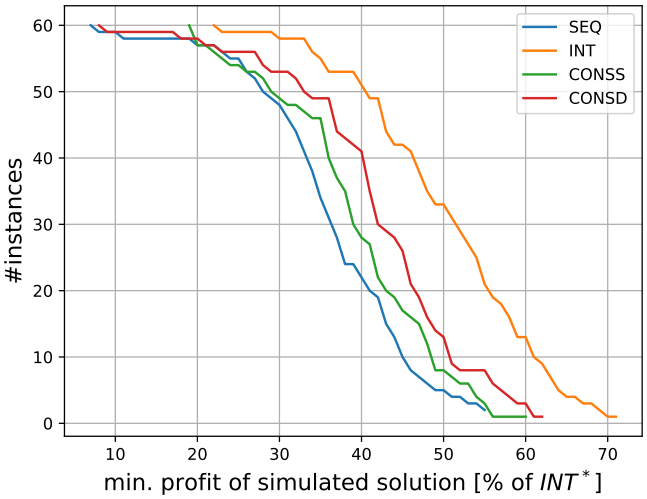
First, we evaluated the solutions found by the four algorithms for each of the 60 instances on the same instance again. As Figure 16a shows, the profit that is then achieved by the integrated solution $INT^\dagger$ ranges from around 20% to just over 70% that of the non-simulated best-known solution INT^*, with more than half of all instances achieving more than 50%. Such degradation is, of course, not altogether surprising, since (a) the assumed user behavior significantly restricts the set of feasible solutions to the problem, thereby worsening the best attainable objective value, and (b) the algorithm does not take these additional restrictions into account during the optimization procedure. However, even if the algorithm were to account for these restrictions, we would, in general, not be able to achieve the same profit as in the unrestricted case due to argument (a).

The sequential algorithm, on the other hand, suffers from an even worse degradation than the integrated one. The profit achievable by its greedily evaluated solutions $SEQ^\dagger$ ranges from less than 10% to 55% that of INT^*, with a median below 40%. This suggests that, in general, the solutions found by our integrated algorithm would fare better in a user-driven car sharing system, even when their behavior is not explicitly taken into account in the optimization procedure.

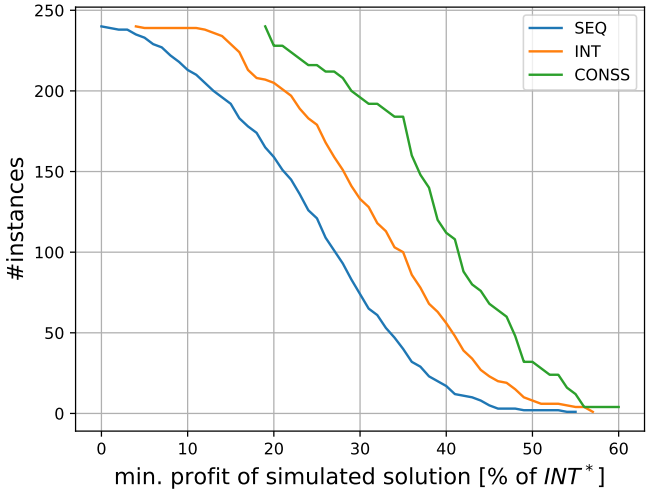
The two greedily evaluated consensus solutions $CONSS^\dagger$ and $CONSD^\dagger$ generally provide a better solution quality than the corresponding sequential solution $SEQ^\dagger$, but are likewise outperformed by the integrated solution $INT^\dagger$ evaluated on its own scenario. This behavior can likely be explained by the fact that they must strike a balance between the different scenarios for which they represent a consensus, whereas algorithm INT can optimize purely for the single scenario under consideration.

During cross-validation, however, the roles of $INT^\dagger$ and the two consensus solutions $CONSS^\dagger$ and $CONSD^\dagger$ are reversed, as can be seen in Figures 16b and 16c, respectively. When INT^* solutions are greedily evaluated on different demand scenarios, their median quality is approximately 30% that of the second instance's INT^* solution, while the corresponding $CONSS^\dagger$'s median quality is around 40%. A similar picture can be seen when INT^* solutions to each instance's day-long subinstances are greedily evaluated on the full week-long instance. Here, the $INT^\dagger$ solutions only reach a median quality of around 25% that of INT^*, while the $CONSD^\dagger$ solutions still achieve over 40% in the median.

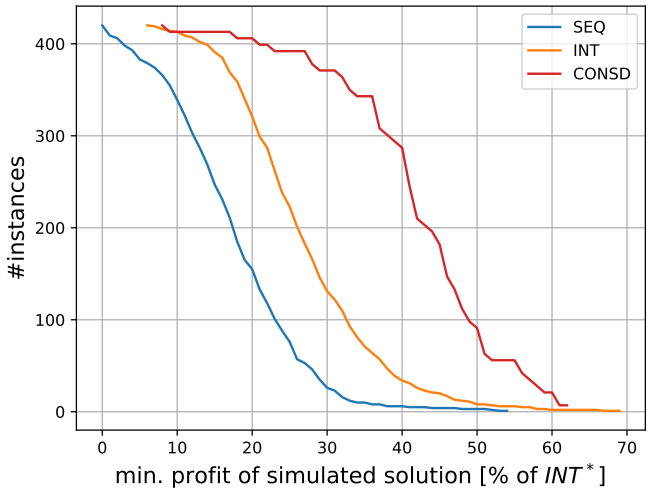
Overall, these results suggest that the relative ordering of the different algorithms with respect to their attainable solution quality does not change when their solutions are greedily evaluated according to expected user behavior instead of optimized from the operator's perspective. When only a single scenario is considered, the integrated algorithm INT still outperforms both the sequential one SEQ and both consensus variants $CONSS$ and $CONSD$ as far as solution quality is concerned. However, once we consider multiple scenarios, the two consensus variants seem to be a better choice than simply using one of the scenarios' INT^* solutions, even if it generally still outperforms the corresponding SEQ^* solution. This suggests that the strategic decisions made by both the integrated and the consensus algorithms are still rather reasonable in a user-driven setting, even if the users' behavior is not explicitly considered during the



(a) greedy evaluation (non-cross-validated)



(b) cross-scenario greedy evaluation



(c) cross-day greedy evaluation

Figure 16: Reverse cumulative sum plots, showing the profit of greedily evaluated solutions vs. the profit of the optimal (or best-known) solution found by our integrated model.

optimization procedure. How much more profit could be gained by incorporating this into the corresponding models and algorithms remains an open question for future research.

9 Conclusion

In this work, we introduced a new optimization problem related to the strategic, long-term planning of an electric car sharing network: the charging station location problem (CSLP). We showed that the problem is NP-hard and remains so if either the available budget or the battery capacity of cars is not restrictive, but can be solved in polynomial time if both are nonrestrictive. We propose two integer linear programming formulations that ensure consistent car movements and three formulations to track each car's battery level, yielding a total of six ILP formulations for the CSLP. A main difference to related problems considered in existing literature is that we explicitly consider and track each car's battery level throughout the planning period. We also described two construction heuristics that are used to quickly derive feasible solutions of good quality to serve as a starting point for our branch-and-cut algorithms.

We evaluated the computational performance of our heuristics and the five most promising exact approaches on two sets of instances: grid graph instances and real-world instances based on data from the city of Vienna. Our experiments showed that the number of potential trips is the principal factor in determining an instance's difficulty and we also analyzed the influence of further parameters. The best formulations were able to solve instances with up to 480 trips to proven optimality, while our heuristics were able to provide high quality solutions with negligible runtime.

In further experiments, we compared the performance of our integrated algorithms that jointly optimize decisions

at various levels to a second sequential approach where these decision levels are optimized separately and showed that they are able to achieve up to 90% more profit than the sequential one. We then analyzed the behavior of our algorithms in cases where demand is uncertain during a cross-validation study and showed that combining the optimal solutions from different deterministic instances into a single consensus solution provides a computationally tractable way to deal with the uncertainty of demand forecasts and possible overfitting of the deterministic model.

Finally, we designed a greedy simulation procedure to evaluate the solutions found by our algorithms in a setting where trip requests are fulfilled on a first-come-first-served basis and customers can freely choose among the available cars and chargers when performing a trip. This simulation study showed that under such a trip selection paradigm, the achievable profit of our solutions can be significantly reduced, since this additional flexibility provided to the customers is not being considered during optimization. Designing models and algorithms that incorporate these aspects thus seems like a promising direction for future research.

Other future work may involve the study of further ILP formulations and corresponding exact algorithms, as well as the development of metaheuristic approaches possibly based on the two heuristics introduced in the current article. We also want to study problem variants that improve the flexibility of the system's users and operators. An example of the former would be a free-floating variant, where users may park cars anywhere within the system's area of operation. For the latter, we want to consider a model where cars may be relocated between busier and less busy zones throughout the planning period, either by the operator's employees or by users. In a system using user-based relocation, users would be incentivized to move cars according to the operator's preference. This can be achieved by employing flexible pricing mechanisms that can be obtained using bi-level optimization.

Finally, we believe that incorporating some aspects of uncertainty into our models would further increase their practical applicability. This includes stochastic variants, e.g., using multiple trip scenarios, as well as ones including some form of robustness, e.g., with respect to the trips' battery consumption or the availability of cars and empty chargers at the opened stations.

Acknowledgements

This work is supported by the Joint Programme Initiative Urban Europe under the grant 847350 and by the Vienna Science and Technology Fund (WWTF) through project ICT15-014. Georg Brandstätter has been supported by a Dissertation Completion Fellowship from the University of Vienna.

The authors thank their project partners from the Austrian Institute of Technology (AIT) for creating the real-world instance from Vienna and for their support in visualizing the obtained results.

The map data we used for our instances and visualizations is copyrighted by OpenStreetMap contributors and available at `http://www.openstreetmap.org/`. The visualizations in Figure 12 were made using Folium (`https://github.com/python-visualization/folium`) and LeafletJS (`https://leafletjs.com/`).

References

Absi N, Archetti C, Dauzère-Pérès S, Feillet D, Speranza MG, 2018 *Comparing sequential and integrated approaches for the production routing problem. European Journal of Operational Research* 269(2):633–646.

Asamer J, Reinthaler M, Ruthmair M, Straub M, Puchinger J, 2016 *Optimizing charging station locations for urban taxi providers. Transportation Research Part A: Policy and Practice* 85:233 – 246.

Baouche F, Billot R, Trigui R, El Faouzi NE, 2014 *Efficient allocation of electric vehicles charging stations: Optimization model and application to a dense urban network. IEEE Intelligent Transportation Systems Magazine* 6(3):33–43.

Barth M, Todd M, 1999 *Simulation model performance analysis of a multiple station shared vehicle system. Transportation Research Part C: Emerging Technologies* 7(4):237–259.

Bektaş T, Laporte G, 2011 *The pollution-routing problem. Transportation Research Part B: Methodological* 45(8):1232 – 1250, supply chain disruption and risk management.

Boyacı B, Zografos KG, Geroliminis N, 2015 *An optimization framework for the development of efficient one-way car-sharing systems. European Journal of Operational Research* 240(3):718–733.

Brandstätter G, Gambella C, Leitner M, Malaguti E, Masini F, Puchinger J, Ruthmair M, Vigo D, 2016 *Overview of optimization problems in electric car-sharing system design and management.* Dawid H, Doerner KF, Feichtinger G, Kort PM, Seidl A, eds., *Dynamic Perspectives on Managerial Decision Making*, volume 22 of *Dynamic Modeling and Econometrics in Economics and Finance*, 441–471 (Springer, Berlin).

Cavadas J, Correia GH, Gouveia J, 2015 *A MIP model for locating slow-charging stations for electric vehicles in urban areas accounting for driver tours. Transportation Research Part E: Logistics and Transportation Review* 75:188–201.

Cepolina EM, Farina A, 2012 *A new shared vehicle system for urban areas. Transportation Research Part C: Emerging Technologies* 21(1):230–243.

Chen TD, Kockelman KM, Khan M, et al., 2013 *The electric vehicle charging station location problem: a parking-based assignment method for seattle. 92nd Annual Meeting of the Transportation Research Board. Washington DC, USA.*

Church R, ReVelle C, 1974 *The maximal covering location problem. Papers of the Regional Science Association*, volume 32, 101–118 (Springer).

Cordeau JF, Furini F, Ljubic I, 2018 *Benders decomposition for very large scale partial set covering and maximal covering problems*. Technical report.

Correia GH, Antunes AP, 2012 *Optimization approach to depot location and trip selection in one-way carsharing systems. Transportation Research Part E: Logistics and Transportation Review* 48(1):233–247.

Correia GH, Jorge DR, Antunes DM, 2014 *The added value of accounting for users' flexibility and information on the potential of a station-based one-way car-sharing system: An application in Lisbon, Portugal. Journal of Intelligent Transportation Systems* 18(3):299–308.

Daimler AG, 2012 *Smart ED booklet*. URL `https://www.smart.com/content/dam/smart/EN/PDF/smart_electric_drive_Broschuere_en_INT.indd.pdf`.

Fassi AE, Awasthi A, Viviani M, 2012 *Evaluation of carsharing network's growth strategies through discrete event simulation. Expert Systems with Applications* 39(8):6692–6705.

Frade I, Ribeiro A, Gonçalves G, Antunes AP, 2011 *Optimal location of charging stations for electric vehicles in a neighborhood in lisbon, Portugal. Transportation Research Record: Journal of the Transportation Research Board* 2252:91–98.

Gallagher M, 1991 *Proportionality, disproportionality and electoral systems. Electoral Studies* 10(1):33 – 51.

Gambella C, Malaguti E, Masini F, Vigo D, 2017 *Optimizing relocation operations in electric car-sharing. Omega* .

Garey MR, Johnson DS, 1978 *"Strong" NP-completeness results: Motivation, examples, and implications. J. ACM* 25(3):499–508.

Gawlik W, Litzlbauer M, Schuster A, Koller H, Reinthaler M, Norman N, Waldbauer M, Bolzer A, Leitner M, 2013 *ZENEM – Zukünftige Energienetze mit Elektromobilität, Endbericht*. Technical report, Technische Universität Wien, Austrian Institute of Technology, Taxi 31300 and Wien Energie, URL `https://www.ea.tuwien.ac.at/fileadmin/t/ea/projekte/ZENEM/ZENEM_829953_publ_Endbericht_final_130919.pdf`.

Ge S, Feng L, Liu H, 2011 *The planning of electric vehicle charging station based on grid partition method. 2011 International Conference on Electrical and Control Engineering (ICECE)* (IEEE).

He L, Mak HY, Rong Y, Shen ZJM, 2017 *Service region design for urban electric vehicle sharing systems. Manufacturing & Service Operations Management* 19(2):309–327.

Hess A, Malandrino F, Reinhardt MB, Casetti C, Hummel KA, Barceló-Ordinas JM, 2012 *Optimal deployment of charging stations for electric vehicular networks. Proceedings of the First Workshop on Urban Networking*, 1–6, UrbaNe '12 (New York, NY, USA: ACM).

Jorge D, Correia GH, 2013 *Carsharing systems demand estimation and defined operations: a literature review. European Journal of Transport and Infrastructure Research* 13(3):201–220.

Kellerer H, 1999 *A polynomial time approximation scheme for the multiple knapsack problem*. Hochbaum DS, Jansen K, Rolim JDP, Sinclair A, eds., *Randomization, Approximation, and Combinatorial Optimization. Algorithms and Techniques*, 51–62 (Berlin, Heidelberg: Springer Berlin Heidelberg).

Laporte G, Nickel S, da Gama FS, 2015 *Location Science*, volume 528 (Springer).

Mitsubishi, 2017 *Mitsubishi iMiEV catalog*. URL `https://www.mitsubishi-motors.com/en/showroom/i-miev/catalog/pdf/17_5my_i_miev_g_exp.pdf`.

Nissan, 2014 *Nissan LEAF owner's manual*. URL `https://owners.nissanusa.com/content/techpub/ManualsAndGuides/LEAF/2014/2014-LEAF-owner-manual.pdf`.

Pelletier S, Jabali O, Laporte G, 2014 *Battery electric vehicles for goods distribution: A survey of vehicle technology, market penetration, incentives and practices*. Technical report, CIRRELT, Montréal, Canada.

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

Ruokokoski M, Solyali O, Cordeau JF, Jans R, Süral H, 2010 *Efficient formulations and a branch-and-cut algorithm for a production-routing problem*. Technical report, GERAD G-2010-66.

Sellmair R, Hamacher T, 2014 *Method of optimization for the infrastructure of charging station for electric taxis. Proceedings of the 93rd Annual Meeting of the Transportation Research Board*.

Shaheen S, Sperling D, Wagner C, 1998 *Carsharing in Europe and North America: Past, present and future. Transportation Quarterly* 52(3):35–52.

Shu J, Song M, 2013 *Dynamic container deployment: two-stage robust model, complexity, and computational results. INFORMS Journal on Computing* 26(1):135–149.

Wang H, Huang Q, Zhang C, Xia A, 2010 *A novel approach for the layout of electric vehicle charging station. The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding* (IEEE).

Wang YW, Lin CC, 2013 *Locating multiple types of recharging stations for battery-powered electric vehicle transport. Transportation Research Part E: Logistics and Transportation Review* 58:76–87.

Worley O, Klabjan D, Sweda TM, 2012 *Simultaneous vehicle routing and charging station siting for commercial electric vehicles. 2012 IEEE International Electric Vehicle Conference* (IEEE).

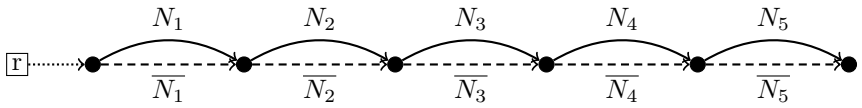


Figure 17: Location graph of a knapsack instance with $N = \{(5,4),(2,3),(7,5),(4,3),(17,8)\}$, $m = 2$ and $c = 10$. Dashed waiting arc $\overline{N_i}$ corresponds to not taking item i, whereas trip arc N_i corresponds to taking it.

A Proofs of Theorem 1, and Lemmas 1 to 3

A.1 Proof of Theorem 1

Proof. We first consider the case of a constant number of potential (i.e., nearby) stations, which is relevant in practice since users will likely only consider a couple of nearby stations. Observe that the number of nodes in G is asymptotically bounded from above by the number of nodes i_t such that i is either a start or end station at time t. This set of nodes is induced by the set of trip arcs A^{T}. Since each trip $k \in K$ is represented by $|N(o_k)| \cdot |N(d_k)|$ trip arcs in G, we have $|A^{\mathrm{T}}| = \mathcal{O}(|K|)$ and thus also $|V| = \mathcal{O}(|K|)$ since neighborhoods are of constant size and since each trip arc may induce at most two nodes. As the maximum number of waiting and initialization arcs may not exceed the number of nodes (no node can be the source or target of more than one waiting or initialization arc) we also have $|A^{\mathrm{I}} \cup A^{\mathrm{W}}| = \mathcal{O}(|K|)$. The claim follows from additionally observing that the number of nodes in G is obviously bounded from above by $\mathcal{O}(|S| \cdot T_{\max})$ since it may consist of at most one node for each station and each considered time point.

To see that, in general, $|V| = \mathcal{O}(|S| \cdot \min\{|K|, T_{\max}\})$ holds, we provide an upper bound on the size of node set V_i for each station $i \in S$. To this end, we observe that there can be at most two nodes associated to station i for each trip (at the trips start and end time). On the other hand, at most one node may correspond to station i for each considered point in time. Thus, we obtain that $|V_i| = \mathcal{O}(\min\{|K|, T_{\max}\})$ and therefore $|V| = \mathcal{O}(|S| \cdot \min\{|K|, T_{\max}\})$. Next, we show that $|A| = \mathcal{O}(|S| \cdot \min\{|K|, T_{\max}\} + |S|^2|K|)$. Recall that the set of arcs is the union of initialization arcs A^{I}, waiting arcs A^{W}, and trip arcs A^{T}. It is immediate that $|A^{\mathrm{I}}| = \mathcal{O}(|S|)$ since there is at most one initialization arc for each station $i \in S$. As mentioned above, each node from $|V|$ can be the source (or target) of at most one waiting arc. Thus, we also have that $|A^{\mathrm{W}}| = \mathcal{O}(|V|) = \mathcal{O}(|S| \cdot \min\{|K|, T_{\max}\})$. Finally, the maximum number of trip arcs is obtained in case each station can be both a start and end station of each trip, in which case A^{T} contains $|S|^2$ arcs per trip. Overall, we obtain $|A| = \mathcal{O}(|S| + |S| \cdot \min\{|K|, T_{\max}\} + |S|^2|K|) = \mathcal{O}(|S| \cdot \min\{|K|, T_{\max}\} + |S|^2|K|)$. □

A.2 Proof of Lemma 1

Proof. We show the NP-hardness of CSLP for this particular case by a reduction from the multiple knapsack problem with identical capacities (MKP-I) (Kellerer 1999).

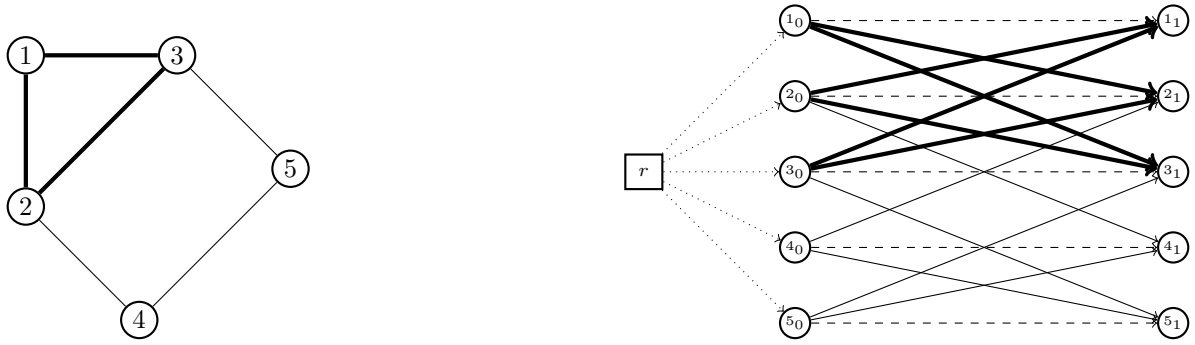
The MKP-I is defined as follows: given m knapsacks, each with capacity $c \in \mathbb{N}$, and n items $N = \{(p_1, w_1), \ldots, (p_n, w_n)\}$, each with associated profit $p_i \in \mathbb{N}$ and weight $w_i \in \mathbb{N}$, the goal is to find a subset of items to pack into the m knapsacks so as to maximize the profit of selected items.

Starting from an arbitrary MKP-I instance, we now define an instance of CSLP as follows:

The input graph $\mathcal{G}$ consists of a single vertex v and no arcs. The set of stations $S = \{v\}$ consists of a single station at vertex v with construction cost $F_v = 0$, per-charger cost $Q_v = 0$, maximum capacity $C_v = m$ and neighborhood $\mathcal{N}_v = \{v\}$. The set of cars consists of m identical cars, each corresponding to one knapsack, with acquisition cost $F_{\mathrm{c}} = 0$, recharge rate $\rho = 0$ (i.e., no recharging is allowed) and maximum battery capacity $B^{\max} = c$ corresponding to the capacity of each knapsack. Note that since all costs related to the opening of stations, the construction of chargers and the purchase of cars are also zero, the instance is budget-unconstrained. Consequently, the budget W can be set to zero.

The planning horizon $T = \{0, \ldots, n\}$ consists of one time period $(i-1, i)$ for each item $i \in N$. Finally, the set of trips $K = \{1, \ldots, n\}$ consists of one trip k for each item $k \in N$ with origin and destination $o_k = d_k = v$, start time $s_k = k - 1$ and end time $e_k = k$ corresponding to the trip's time period, as well as battery consumption $b_k = w_k$ and profit p_k according to the corresponding item's weight and profit, respectively. The resulting location graph of an example instance can be seen in Figure 17.

Solutions of the multiple knapsack instance and our instance are now linked with the following simple equivalence: each of m knapsacks corresponds an available car, each item is a possible trip, an item is added to a knapsack if and only if its corresponding trip is accepted and assigned to that knapsack's corresponding car. By construction, any feasible CSLP solution corresponds to a feasible MKP-I solution of the same cost. Hence, the equivalence between the two problems follows immediately. □



(a) Original graph (b) Location graph

Figure 18: Example instance of the clique problem with 5 vertices and its corresponding location graph. Edges between the vertices in the maximum clique $C = \{1, 2, 3\}$, as well as their corresponding trip arcs in the location graph, are highlighted in bold.

A.3 Proof of Lemma 2

Proof. We will prove this lemma by reduction from the clique problem (Garey and Johnson 1978). Given an undirected graph $G' = (V', E')$ and an integer $\kappa \geq 2$ the clique problem consists of finding a subset of vertices $C \in V'$ of cardinality κ such that the induced subgraph $G'[C]$ is complete.

Given an arbitrary instance of the clique problem, we define an instance of CSLP as follows (see Figure 18 for an example): The input graph $\mathcal{G}$ is a complete graph containing one vertex for each vertex in G'. The set of charging stations $S = V'$ contains all vertices from $\mathcal{G}$, each with opening cost $F_i = 1$, capacity $C_i = \infty$ and per-charger cost $Q_i = 0$. The number of available cars H is chosen sufficiently large and each car has acquisition cost $F_c = 0$, battery capacity $B^{\max} = \infty$ and recharge rate $\rho = 0$ (i.e., no recharging is needed). The budget is $W = \kappa$ and the planning period $T = \{0, 1\}$ consists of two time periods. For each edge $\{i, j\} \in E'$, we add two trips to our instance: one from i to j and one from j to i, with the stations at these vertices as their only start and end station. Each trip starts at $s_k = 0$, ends at $e_k = 1$, has battery consumption $b_k = 0$ and profit $p_k = 1$. Note that since every trip consumes zero battery, the instance is battery-unconstrained.

Given these assumptions, it is not difficult to see that the optimal solution of the CSLP on the transformed graph is equal to $p = 2 \cdot \binom{\kappa}{2}$ if and only if the selected stations in this solution correspond to a clique of size κ in G'. Indeed, at most κ stations can be opened, and the profit associated to a trip from i to j can be collected only if both stations i and j are opened. Hence, the maximum total profit that can be collected by opening κ stations is $p = 2 \cdot \binom{\kappa}{2}$, and given that two trips (one from i to j and one from j to i) in $\mathcal{G}$ correspond to an edge in G', this profit can be achieved only if a clique of size κ exists in G, which concludes the proof. □

A.4 Proof of Lemma 3

Proof. We will prove this result by showing how to obtain an optimal solution to a budget- and battery-unconstrained instance of CSLP from a minimum-cost flow on the location flow network.

Let $\tilde{f}_a \geq 0$ denote the flow value of each arc $a \in \tilde{A}$ associated to a minimum-cost flow on location flow network $\tilde{G}$ with associated objective value $Z^* = \sum_{a \in \tilde{A}} c_a \tilde{f}_a = -\sum_{a=(v_k, v'_k) \in \tilde{A} | k \in K, \tilde{f}_a = 1} p_k$. Let furthermore, I denote a budget- and battery-unconstrained instance of CSLP such that $\tilde{G}$ is the associated location flow network. To obtain a solution $(S', \bigcup_{i \in S} C'_i, H', K', \mathcal{H}', \text{start}', \text{end}')$ of I, all stations are opened at their maximum capacity (i.e., $S' = S$ and $C'_i = C_i$, $\forall i \in S$) and one car is purchased for each unit of flow leaving the root node that is not routed to the sink node via arc (r, s), i.e., $H' = \sum_{a \in \delta^+(r) \setminus \{(r,s)\}} \tilde{f}_a$. Trips $k \in K$ are accepted (and thus included in K') if and only if the flow $\tilde{f}_a$ associated to the corresponding trip arc $a = (v_k, v'_k) \in \tilde{A}$ is equal to one. Note, that the chosen capacities $\varsigma_a = 1$ together with the fact that the all demand and capacities are integral ensure that the flow along each such trip arc is either zero or one and hence we obtain a solution with objective value $-Z^*$. For each selected trip $k \in K'$ we further observe that the flow values of exactly one start station arc (i'_{s_k}, v_k) and one end station arc (v'_k, j_{e_k}) will be equal to one and therefore define the selected start and end stations $\text{start}(k) = i$ and $\text{end}(k) = j$, respectively. Nonzero flow values $\tilde{f}_a$ on root arcs (r, i_t) correspond to an initial allocation of $\tilde{f}_a$ cars at station i and on waiting arcs $a = (i_t, i'_t)$ indicate that $\tilde{f}_a$ cars are waiting at station i between time points t and t'. We observe that the capacities imposed on root and waiting arcs ensure that the capacities of stations are never exceeded.

It remains to show that each purchased car performs a connected route through open stations of the street network such that all selected trips are covered by one car. To this end, we observe that due to the acyclicity of $\tilde{G}$, flow $\tilde{\boldsymbol{f}}$ can be easily decomposed into a set of paths (from r to s), each of which imposed by one unit of flow ($\text{car}(k)$ is set accordingly for each arc corresponding to a trip $k \in K$). Each such path represents the route of a purchased car

through the set of open stations and it is immediate from the definition of $\tilde{G}$ that each path from r to s is a connected route through the stations of the street network.

The fact that the solution is optimal can be easily shown by contradiction: suppose there were another solution with higher total profit. Then, by assigning the flow variables on the location flow graph according to the procedure described above, we would be able to find a feasible flow with less cost, which contradicts our assumption that the original flow was a minimum-cost one. □

B Detailed description of the sequential algorithm

Using the integer linear programming formulation (44)–(50), the sequential algorithm first solves a variant of the maximal covering location problem (which was introduced by Church and ReVelle (1974); for further information, see also Laporte, Nickel, and da Gama (2015), Cordeau, Furini, and Ljubic (2018)) to determine which charging stations to open. Binary variables $y_i,\ \forall i \in S$, take value 1 if the corresponding station should be opened, and value 0 if they should remain closed. Start station assignment variables $x_{ki},\ \forall k \in K,\ i \in N(o_k)$, are 1 if trip k is assigned to station i as a start station, and 0 otherwise. Likewise, end station assignment variables $x'_{ki},\ \forall k \in K,\ i \in N(d_k)$ are used to assign end stations to each accepted trip.

$$\max \sum_{k \in K} \left(\sum_{i \in N(o_k)} p_k x_{ki} + \sum_{i \in N(d_k)} p_k x'_{ki} \right) \tag{44}$$

$$\text{s.t.} \sum_{i \in S} (F_i + Q_i \cdot C_i) \cdot y_i \leq W - F_c \cdot H \tag{45}$$

$$\sum_{i \in N(o_k)} x_{ki} \leq 1 \qquad \forall k \in K \tag{46}$$

$$\sum_{i \in N(d_k)} x'_{ki} \leq 1 \qquad \forall k \in K \tag{47}$$

$$x_{ki} \leq y_i \qquad \forall k \in K,\ i \in N(o_k) \tag{48}$$

$$x'_{ki} \leq y_i \qquad \forall k \in K,\ i \in N(d_k) \tag{49}$$

$$\sum_{i \in N(o_k)} x_{ki} = \sum_{i \in N(d_k)} x'_{ki} \qquad \forall k \in K \tag{50}$$

The objective function (44) maximizes the profit obtained from all trip starts and end that could be assigned to a station. Note that each trip's profit is counted twice here (once for the start, once for the end station assignment). Since we are only concerned with the ultimate values of the y_i variables, this profit overestimation by a factor of two does not change the algorithm's outcome. However, in case a realistic profit estimation according to this model would be required, objective function coefficients $\frac{p_k}{2}$ should be used instead. Budget constraint (45) ensures that the expenditures for opening stations remain within the given budget limit, while also ensuring that sufficient money is left to buy vehicles (since that decision is made at the second level). Assignment constraints (46) and (47) guarantee that each trip can be assigned at most one start and end station, while linking constraints (48) and (49) ensure that in order for a trip to be assigned to a station (either as a start or end station), that station must be opened. Finally, trip balance constraints (50) ensure that a trip can only be accepted (and thus, its profit collected) if it is assigned both a start and an end station. This represents the directionality of demand.

After the station locations have been determined by the previous phase, the remaining decisions are optimized with algorithm FC (see Sections 4 and 5) where we fix the values of variables y_i and z_i to 1 and C_i for those stations that were opened, and to 0 and 0 for those that were not. Thus, in this phase, the size of our vehicle fleet, as well as the assignment of trips to and the routing of these vehicles are determined.

C Stochastic formulation for obtaining consensus solutions

Let Ω be a discrete set of demand scenarios and $K_\omega,\ \omega \in \Omega$ denote the set of trips in each of the scenarios. Together with the remaining input, these scenarios induce a set of $|\Omega|$ (single-scenario) CSLP instances that differ with respect to their trip requests but share all remaining instance parameters (e.g., the underlying street network, the set of potential charging stations and their parameters, and the vehicles' parameters).

First, each of these instances is solved by our integrated algorithm INT. The solutions found in this phase (specifically, their values for variables $y_i,\ z_i,\ i \in S$ and $x_k,\ k \in K_\omega$) are then used in the second phase to reduce the search space where we build a restricted two-stage stochastic ILP formulation (51)–(65) for scenario set Ω.

$$\max \sum_{\omega\in\Omega} P_\omega \cdot \sum_{k\in K_\omega} p_k x_k \tag{51}$$

$$\text{s.t.} \sum_{i\in S}(F_i y_i + Q_i z_i) + \sum_{h=1}^{H} F_c a_h \le W \tag{52}$$

$$z_i \le C_i y_i \qquad \forall i \in S \tag{53}$$

$$\sum_{h=1}^{H} x_k^h = x_k \qquad \forall \omega \in \Omega,\ k \in K_\omega \tag{54}$$

$$\sum_{k\in K_\omega : s_k \le t, e_k > t} x_k^h \le a_h \qquad \forall \omega \in \Omega,\ h \in \{1,\dots,H\},\ t \in \bigcup_{i\in S} T_i^+ \tag{55}$$

$$\sum_{h=1}^{H} f^h[\delta^-(i_t^\omega)] \le z_i \qquad \forall \omega \in \Omega,\ \forall i \in S, \forall t \in \left(T_i^- \cup \{t' \mid \exists (r, i_{t'}) \in A_\omega\}\right) \tag{56}$$

$$f^h[\delta^+(r_\omega)] = a_h \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\} \tag{57}$$

$$f^h[\delta^-(i_t^\omega)] - f^h[\delta^+(i_t^\omega)] = 0 \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\}, \forall i_t \in V, \exists (i_t, j_{t'}) \in A : t' > t \tag{58}$$

$$f^h[\delta^-(i_t^\omega)] - f^h[\delta^+(i_t^\omega)] \ge 0 \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\},\ \forall i \in S,\ t = \max(T_i^+ \cup T_i^-) \tag{59}$$

$$\sum_{a\in A_{\mathrm{T}}^k} f_a^h = x_k^h \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\},\ \forall k \in K_\omega \tag{60}$$

$$f_a^h \in \{0,1\} \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\},\ \forall a \in A_\omega \tag{61}$$

$$g_{e_k}^{h\omega} - g_{s_k}^{h\omega} \le -b_k x_k^h + \Delta_k \rho (1 - x_k^h) \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\},\ \forall k \in K_\omega \tag{62}$$

$$g_t^{h\omega} - g_{\pi(t)}^{h\omega} \le \rho(t - \pi(t)) a_h \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\},\ \forall t \in T^{\mathrm{B}} \setminus \{0\} \tag{63}$$

$$g_0^{h\omega} = B^{\max} \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\} \tag{64}$$

$$0 \le g_t^{h\omega} \le B^{\max} \qquad \forall \omega \in \Omega,\ \forall h \in \{1,2,\dots,H\},\ \forall t \in T^{\mathrm{B}} \setminus \{0\} \tag{65}$$

Since formulation (51)–(65) is a straightforward extension of the single-scenario ILP described in Section 4 we refrain from describing all constraints in detail. Note, however, that each scenario $\omega \in \Omega$ induces an individual time-expanded location graph whose nodes are referred to as r_ω and i_t^ω, respectively. The first-stage decisions are typical strategic ones: they consist of station opening and capacity variables $y_i,\ z_i,\ i \in S$, as well as vehicle acquisition variables $a_h,\ h \in \{1,\dots,H\}$. The second-stage decisions for each scenario then consist of the remaining operational ones, namely the selection of trips to accept, their assignment to vehicles and their routing.

To obtain a consensus solution, we restrict this model by setting the variable bounds of variables $y_i,\ z_i,\ i \in S$ and $x_k,\ \omega \in \Omega,\ k \in K_\omega$ according to the single-instance solutions INT^*. More precisely, the lower bound of each variable y_i is set to the minimum y_i value taken in any of those solutions, while its upper bound is set to the corresponding maximum y_i value across all given solutions. The lower and upper bound of the z_i variables are similarly set based on the minimum and maximum values these variables take in the single-scenario solutions. Finally, the upper bound of each variable x_k is set to zero whenever trip k was not accepted in the solution to the single-instance scenario containing k.

D Detailed description of the greedy simulation algorithm

Algorithm 2 gives a detailed description of the simulation procedure. Following the terminology established in Section 2, it takes a set of charging stations S' that are opened, as well as a vector $\boldsymbol{C'}$ indicating the number of chargers built at each station (closed stations always have zero chargers), and outputs the set of trips that the operator will obtain with this setup, assuming all customers behave as described in Section 8. As in Algorithm 1, we use W' to indicate the remaining budget. Multisets B_{it} contains the battery levels of all cars parked at station i at time t, while K_{acc} collects the trips that were accepted.

First, we remove all trips from K that can no longer be accepted since all their potential start or end stations are closed. Next, we distribute as many vehicles as possible among the open stations. To ensure that cars are placed close to where we expect our customers to need them, we use D'Hondt's method that is commonly used in parliamentary elections to allocate seats to party lists based on the share of votes they received during the election (see, e.g., Gallagher (1991)). In lieu of votes, we use the profit potential p_i of each open station, i.e., the sum of the profit of all trips for which it serves as a potential start or end station. Fully charged cars are then placed at the station with the highest quotient between its profit potential and the number of cars already parked there (plus one)

$K' = \{k \in K \mid N(o_k) \cap S' \neq \emptyset,\ N(d_k) \cap S' \neq \emptyset\}$
$B_{it} = \emptyset,\ \forall i \in S,\ t \in T$
$W' = W - \sum_{i \in S'} F_i - Q_i \cdot C'_i$
$K_{\text{acc}} = \emptyset$
// allocate cars based on D'Hondt's method
for $i \in S'$ **do**
  $p_i = \sum_{k \in K' : i \in N(o_k)} p_k + \sum_{k \in K' : i \in N(d_k)} p_k$
for $h \in \{1, \ldots, H\}$ **do**
  if $W' \geq F_{\text{c}}$ **then**
    for $i \in S'$ **do**
      $quot_i = \frac{p_i}{|B_{i0}|+1}$
    $j = \arg\max_{i \in S' : |B_{i0}| < C'_i} \{quot_i\}$ // obey capacity limit
    if $j = \emptyset$ **then break**
    $B_{j0} = B_{j0} \cup \{B^{\max}\}$
    $W' = W' - F_{\text{c}}$
// simulation phase
for $t \in T \setminus T_{\max}$ **do** // in ascending order
  // perform trips starting at t
  for $k \in K' : s_k = t$ **do**
    for $i \in N(o_k) \cap S'$ **do** // in order of preference
      for $j \in N(d_k) \cap S'$ **do** // in order of preference
        if $\max\{B_{it}\} \geq b_k \wedge |B_{je_k}| < C'_j$ **then**
          $B_{je_k} = B_{je_k} \cup \{\max\{B_{it}\} - b_k\}$
          $B_{it} = B_{it} \setminus \max\{B_{it}\}$
          $K_{\text{acc}} = K_{\text{acc}} \cup \{k\}$
          goto next trip
  // leave and charge unused cars at their current station
  for $i \in S'$ **do**
    $B_{i,t+1} = \{\min\{b + \rho, B^{\max}\} \mid b \in B_{it}\}$
    $B_{it} = \emptyset$

Algorithm 2: Greedy simulation algorithm

until we run out of available cars, budget, or available chargers to place them at. During the simulation phase, we go through the planning horizon T in temporal order. First, for each trip k starting at time t, we try to find the most-preferred station from which it can start (i.e., the closest within walking distance of its origin o_k that has a car with sufficient battery of at least b_k available), as well as the most-preferred station at which it can end (i.e., the closest within walking distance of its destination d_k that has a free charger available). If such stations exist, we accept the trip request with these two stations as its start and end station, respectively, and move the car with the highest battery level accordingly. Otherwise, we decline it. Once all trip requests starting at t have either been accepted or declined, we move all currently parked cars forward by one time period, recharging their battery in the progress. When the algorithm finishes, set K_{acc} contains all trip requests that were accepted under the previously described first-come-first-served paradigm.