# Optimizing Charging Station Locations for Electric Car-Sharing Systems[⋆]

Benjamin Biesinger, Bin Hu, Martin Stubenschrott, Ulrike Ritzinger, and Matthias Prandtstetter

AIT Austrian Institute of Technology
Mobility Department - Dynamic Transportation Systems
Giefinggasse 2, 1210 Vienna, Austria.

⟨firstName⟩.⟨lastName⟩@ait.ac.at

**Abstract.** This paper is about strategic decisions required for running an urban station-based electric car-sharing system. In such a system, users can rent and return publicly available electric cars from charging stations. We approach the problem of deciding on the location and size of these stations and on the total number of cars in such a system using a bi-level model. The first level of the model identifies the number of rental stations, the number of slots at each station, and the total number of cars to be acquired. Then, such a generated solution is evaluated by computing which trips can be accepted by the system using a path-based heuristic on a time-expanded location network. This path-based heuristic iteratively finds paths for the cars through this network. We compare three different pathfinder methods, which are all based on the concept of tree search using a greedy criterion. The algorithm is evaluated on a set of benchmark instances which are based on real-world data from Vienna, Austria using a demand model derived from taxi data of about 3500 taxis operating in Vienna. Computational tests showed that for smaller instances the algorithm is able to find near optimal solutions and that it scales well for larger instances.

**Keywords:** location problem, car-sharing, electric cars, variable neighborhood search

## 1 Introduction

Urban transportation as a major consumer of energy and contributor to air pollution raises challenges to local governments and companies that must be solved in the near future. One of those challenges is to reduce the usage of conventional cars in urban areas. It can already be observed that the market for alternatives to combustion engine powered vehicles, especially (full) battery electric cars, is

[⋆] This work has been partially funded by the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit) in the JPI Urban Europe programme under grant number 847350 (e4share).

steadily growing. Moreover, car-sharing systems as alternative or addition to public transport is getting increasingly popular and in many larger cities such systems are already installed and in use. As the main drawbacks of electric cars—high acquisition costs, limited battery range, and high infrastructure costs—is mitigated when used in an urban shared environment, electric car-sharing systems attracted increased attention over the last years. Such car-sharing systems provide a fleet of vehicles in a defined area of operation in which users can rent and return cars. Specifically for electric cars, however, charging stations have to be installed to recharge the battery of the vehicles. We assume a one-way station-based system in which the users can rent and return the cars only at these stations in contrast to free-floating systems where the customers can return the cars in any free parking space within the operational area. Therefore, we consider stationary electric car-sharing systems where the decision is where to place these stations and how many charging slots to install. This is crucial for the functionality and success of the whole system. A good location for a station naturally depends on the customer demand of the nearby area and therefore we use a demand model to evaluate station locations. The demand model is given by a forecast of a set of trip requests in which each request has an individual estimated profit. The evaluation method maximizes the total profit of the acceptable requests.

This work considers the problem of deciding the charging locations, their size, and the total number of cars in the system as combinatorial optimization problem which is heuristically solved using a two-stage solution algorithm. In the first stage, the decision variables are fixed using a variable neighborhood search (VNS) [12]. Each generated solution candidate is evaluated in the second stage using an iterated procedure based on a greedy criterion using a demand forecast. Specifically, a greedy, a PILOT, and a beam search algorithm is used to iteratively find paths for the used cars through the system. The presented algorithm is evaluated on a set of benchmark instances based on real-world data of Vienna, Austria. First, the problem is formally defined in Section 2. Then, related work is described in Section 3 followed by the description of the algorithm in Section 4. Computational results are presented in Section 5 and finally, conclusions are drawn and possible future research directions are given in Section 6.

## 2 Problem Definition

The charging station location problem (CSLP) is defined on a road network $G = (V, A)$ given by a set of vertices $V$ and a set of directed arcs $A$ representing road segments. Each arc $a = (i, j) \in A$, $i, j \in V$ has a length $l_{ij}$ and an associated weight representing the travel time $\delta_{ij}$ needed to travel from vertex $i$ to $j$. Possible charging stations $S \subseteq V$ are given by a subset of the vertices and each potential station $i \in S$ has an associated opening cost $F_i \geq 0$, a capacity $C_i \in \mathbb{N}$, and a cost per slot $Q_i \geq 0$. The maximum number of cars is given by

$H$, and each car has the same acquisition cost $F_c$, battery capacity $B^{\max}$, and charging rate per time unit $\rho$.

Furthermore, a set of requested trips $K$ is given, which corresponds to a demand forecast within the time horizon $T = \{0, \ldots, T_{\max}\}$ and is the basis of the solution evaluation. Each request $k \in K$ has a starting $s_k \in T$ and ending time $e_k \in T$ with $e_k > s_k$, an origin $o_k \in V$, and a destination $d_k \in V$. Further, a duration $\delta_k$, an estimated battery consumption $b_k$, and a profit $p_k$ is given for each request $k \in K$. It is assumed that the customers are willing to walk to a nearby station if the walking time does not exceed a pre-defined maximum duration $\beta^w$. Based on this maximum walking time we have a set of potential starting $N(o_k)$ and ending stations $N(d_k)$ for each request $k \in K$. If there are no potential stations within the walking range, i.e., $N(o_k) = \emptyset$ or $N(d_k) = \emptyset$ then the request is unrealizable and therefore not considered.

The goal of the CSLP is to find the set of stations to open $S' \subseteq S$, the number of slots to use for each open station, and the total number of cars $H' \leq H$ in the system with a limited budget $W$ such that the total profit of all accepted trips is maximized. To compute the obtained profit each car $c = 0, \ldots, H' - 1$ must be assigned a set of feasible trips $K'_c \subseteq K$ where each trip $k \in K'_c, \forall c = 0, \ldots, H' - 1$ can only be assigned to at most one car $c$. Each car $c = 0 \ldots, H' - 1$ must be able to perform its assigned trips $K'_c$ by ensuring *capacity feasibility*, *battery feasibility*, and *connectivity* of the car route:

- *Capacity feasibility* is given when at each time-step $t \in T$ there are no more cars in station $s \in S$ than the available number of slots.
- *Battery feasibility* is given if the battery capacity of the car is sufficient for performing the requested trip taking potential preceding battery charging into account. More formally, the solution is battery feasible if between two consecutive trips $k^1, k^2 \in K$ starting / ending at station $i$ of a car $\min\{(s_{k^2} - e_{k^1})\rho + B^{k^1}, B^{\max}\} \geq b_{k^2}$ is valid, where $B^{k^1}$ is the remaining battery capacity of the car after performing trip $k^1$.
- *Connectivity* is given when the ending station of a trip $k$ is equal to the starting station of the next trip.

Then, the total profit of a solution $S$ is the total sum over the profits of all accepted trips: $p(S) = \sum_{c=0}^{H'-1} \sum_{k \in K'_c} p_k$. We are aware that in practice it might be unrealistic to plan the accepted trips in such a way, but this academic problem formulation allows us to obtain an upper bound for the total profit based on a given demand forecast.

## 3 Related Work

The charging station location problem in the presented form is a relatively new research topic and was introduced by Brandtstätter et al. [5]. In their work the authors defined the problem, proved its NP-hardness and described some polynomially solvable cases. Furthermore, they modeled the CSLP as integer linear program in several ways and compared the efficiency of their models on a set of artificially created and real-world instances. The best formulations are able to solve real-world instances with up to 480 trips and 50 cars to proven optimality. In another work Brandstätter et al. [4] considered stochastic aspects of the demand and presented two-stage stochastic models and a heuristic for solving the stochastic CSLP.

The literature of optimization problems arising in (electric) car-sharing systems is, however, broader and a recent literature survey can be found in [3]. A similar problem concerning the location of charging stations for electric car-sharing systems was described by Boyacı et al. [2]. Additionally, they also considered operational decisions regarding relocation of cars from stations in low to stations in high demand areas to balance the system as a whole. The topic of relocating cars is also considered by Weikl and Bogenberger [15] who investigated different relocation strategies for free floating conventional car-sharing systems. Related problems in the domain of exact methods and heuristics for optimization problems arising in electric car-sharing systems are described, e.g., by Hess et al. [11], Ge et al. [10], Cavadas et al. [6], and Frade et al. [8]. Another related problem of placing charging stations for electric taxis in urban areas was approached by Asamer et al. [1]. The authors propose a decision support system which identifies promising regions in which charging stations should be placed instead of actual locations.

## 4 Algorithm Description

We approach this problem using a two-stage solution algorithm. In the first (or upper) level it is fixed which stations are opened, how many slots are built at each open station, and how many cars are purchased. Then, in the second (or lower) level we compute which requests can be accepted by the system using the solution of the upper level problem. As both the upper level and the lower level problem are computationally demanding, we solve both of them heuristically. Therefore, a variable neighborhood search (VNS) [12] is used for the upper level and three different procedures based on a greedy criterion are used for solving the second stage. In the next sections we will describe the algorithms in detail.

### 4.1 Variable Neighborhood Search for the Upper Level Problem

A solution to the upper level problem is represented by an integer vector $z$ of size $|S|$ and each element $z_i$, $\forall i = 0, \dots, |S| - 1$ can take values $0, \dots, C_i$. In our approach we determine the total number of cars of a solution candidate implicitly by first fixing the stations and number of slots and then using as many cars as possible with the remaining budget bounded by $H$.

The first step of the algorithm is to create an initial solution which is then passed on to a subsequent variable neighborhood search. The initial solution generation is a greedy construction heuristic based on following greedy criterion: Each station is assigned an *attractiveness* value which determines the order of the stations which are considered to be opened. This attractiveness value is the number of requests which can start or end at this station and thereby represents a heuristic guidance which stations are useful for the given set of requests. Then, the construction algorithm iterates over the set of stations in descending order w.r.t. their attractiveness values, opens station $i$ if the remaining budget is sufficient, and chooses the number of slots out of the budget feasible values of $\{1, \dots, C_i\}$ uniformly at random. The algorithm ensures that there is enough remaining budget to purchase at least one car and terminates if no station can be opened anymore.

This starting solution is then passed to a general variable neighborhood search, which was introduced by Hansen and Mladenović [12]. The underlying variable neighborhood descent (VND) uses four different neighborhood structures (NBs) which are searched in the following order in a best improvement fashion:

- *Close station:* In this neighborhood structure a randomly chosen station out of the currently open ones is closed. As a consequence, this move primarily increases the number of purchased cars and can therefore improve the objective value. Here, we do not only accept improving moves but also accept moves which do not change the objective value. The idea behind this strategy is to have as much budget available for the next neighborhood structures as possible.
- *Open station:* A move in this neighborhood structure opens a previously closed station with as many slots as possible.
- *Change slots:* An already open station $i$ is chosen, the number of slots $z_i$ is set to all feasible values out of $\{1, \dots, C_i\}$ and the best result is taken.
- *Swap slots:* For each pair of stations $i, j \in S$ for which $z_i \neq z_j$ the values of $z_i$ and $z_j$ are swapped. Resulting infeasibilities are repaired by iteratively reducing the number of slots of station $i$ or $j$ (chosen uniformly at random) by one until the solution is feasible again.

The VNS uses four shaking neighborhood structures which are based on the NBs described above. The first two shaking NBs are based on the *swap slots* neighborhood structure and performs two and four swaps, respectively. The third and fourth NBs perform two and four moves in the *close station* neighborhood structure.

## 4.2 Path-based Heuristic

After fixing the solution $(z, H')$ of the upper level problem, in the lower level problem we now have to compute the trips that are accepted by the system so that the total profit is maximized. Therefore, we use a variant of the path-based heuristic (PBH) introduced by Brandstätter et al. [5]. The PBH finds iteratively paths for each car through space and time in an adaptable time-expanded location network (TELN). The TELN is a time-discretized directed acyclic multigraph basically consisting of one node for each station and time unit and arcs between each two consecutive time slots. In the following we will formally define the TELN $G^{\mathrm{T}} = (N, A')$:

First, the set of open stations is defined as $S' = \{i \in S \mid z_i > 0\}$. Then, for each station $i \in S'$ and each time unit $t \in T$ a node $i_t$ is introduced forming the set of nodes $N$. Additionally an artificial source $r^s$ and target node $r^t$ are created. The set of arcs $A'$ is divided into three disjoint subsets of arcs $A^I$, $A^W$, $A^T$ and each arc $a$ has an associated profit $p_a$ and a battery consumption $b_a$.

– The set of *initialization arcs* $A^I = \{(r^s, i_0) \mid i \in S'\} \cup \{(i_{t_{\max}}, r^t) \mid i \in S'\}$ are the arcs leaving the artificial source node to each station node of the first time slot and entering the artificial target node from each station node of the last time slot $t_{\max}$. All initialization arcs have a profit and battery consumption of zero.
– *Waiting arcs* $A^W$ are added between two time slots of any station, i.e., in the simplest case $A^W = \{(i_t, i_{t'}) \mid i \in S', t \in T \setminus t_{\max}, t' = t + 1\}$. This set, however, can be reduced as we will see later. These arcs also have a profit of zero but a negative battery consumption of $-\phi_i$ which corresponds to battery charging.
– Finally, the set of *trip arcs* $A^T$ corresponds to the requested trips and contains arcs for each trip that can be accepted (denoted by $K' \subseteq K$) with the given set of stations $S'$. For each such trip $k \in K'$ an arc is introduced for all possible starting and ending station combinations, i.e., $A^T = \bigcup_{k \in K'} A^T_k$, with $A^T_k = \{(i_{s_k}, j_{e_k}) \mid i \in N(o_k), j \in N(d_k)\}, \forall k \in K'$. Each trip arc $a$ has battery consumption $b_k$ and profit $p_k$, where $k \in K'$ is the corresponding request.

An illustrative example of a TELN with four stations is shown in Figure 1. In this figure the solid lines are initialization or waiting arcs and the dashed lines are trip arcs corresponding to a request.

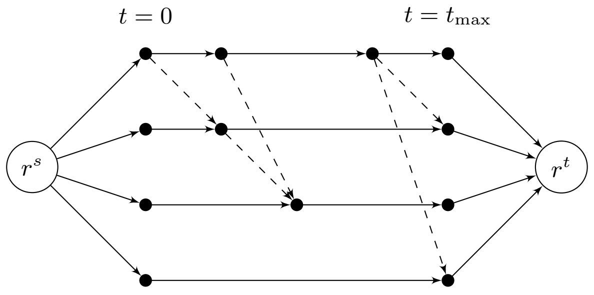

**Fig. 1.** Example of a time-expanded location network with four open stations.

In this example we see a reduced version of the graph which omits many of the unnecessary waiting arcs. Let $N^i \subseteq N = \{n^i_0, \ldots, n^i_{|N_i|}\}$ be the set of all starting / ending nodes of all trips at station $i \in S'$ sorted in ascending order of their starting / ending time. Then, the set of actual introduced waiting arcs is the following: $A^W = \{(i_0, n^i_0)\} \cup \{(n^i_{|N_i|}, i_{t_{\max}})\} \cup \{(n^i_t, n^i_{t+1}) \mid t = 0, \ldots, |N_i| - 1\}$.

After building the graph $G^{\mathrm{T}}$, as Algorithm 1 shows, the PBH consists of iteratively finding paths from $r^s$ to $r^t$ in the TELN and updating the graph based on the found path. The resulting objective value of the upper level solution is then the total profit of all found paths where the profit of a path is defined as the total profit of all requests fulfilled by this path. The path finding step corresponds to solving a resource constrained shortest path problem with the battery as resource and the negative profits as the arc weights to minimize. Note that here we do not have to consider the station capacities as it is assured by the update procedure of the graph, which is described later, that they can never be violated. As this problem is NP-hard [9], we use several heuristics for solving it. All of these heuristics construct iteratively a path starting from $r^s$ using a greedy criterion based on the concept of a *potential profit* $p'$ of a node. For each node $i_t \in N$ a potential profit $p'(i_t)$ is calculated by computing shortest paths from $i_t$ to the target node $r^t$ relaxing the battery constraints. In the PBH this is done by computing shortest paths from $r^t$ to all other nodes in the arc-reversed graph. Then, each time a path $P$ is found through the TELN, the graph is updated for the next iteration. Therefore, for each served request $k \in K'$ on $P$, all corresponding trip arcs $A_k^T$ are deleted from $G^{\mathrm{T}}$ which ensures that one request can only be fulfilled by at most one car. Furthermore, we have to ensure capacity feasibility of the paths and therefore we introduce global variables $u_{it}$, $\forall i \in S', t \in T$ which are initially set to zero. These variables store the current number of cars at each station in each time step over all iterations. For each used arc $a = (i_t, j_{t'}) \in P$, $i, j \in S'$, we increase $u_{jt'}$ by one and check if the fixed number of slots of $j$ is reached. If $u_{jt'} = z_j$, then all incoming arcs of node $j_{t'}$ are deleted from $G^{\mathrm{T}}$ which ensures that in later iterations this node cannot be reached anymore. Finally, also the potential profits of the nodes have to updated and the shortest path values are recomputed.

```
Algorithm 1: Path-based Heuristic
  Input : Upper level solution $(z, H')$
  Output: Approximated profit
  build time-expanded location graph $G^{\mathrm{T}}$
  for $i = 0, \ldots, H'$ do
      find feasible path from $r^s$ to $r^t$ in $G^{\mathrm{T}}$
      update graph $G^{\mathrm{T}}$
      if termination criterion is satisfied then
          break;
  return total profit of all paths
```

The PBH terminates if one of these conditions is satisfied:

- $H'$ paths are found.
- The potential profit of $r^s$ is zero or $r^t$ is unreachable.

– The path finding algorithm could not find a path from $r^s$ to $r^t$. This can happen if it gets stuck at a node with either no outgoing arc or with only outgoing arcs with a battery consumption greater than the remaining capacity.
– The profit of the found path is zero. Although this condition is not needed for the PBH to be valid it is introduced for improving its efficiency by reducing the number of generated paths with zero profit.

In the following we will describe three heuristics we designed, implemented, and evaluated for finding feasible paths. They are described in order of their complexity and time consumption and a comparison of their performance is given in Section 5.3.

**Greedy pathfinder** The greedy pathfinder always appends at the current node $i_t$ a battery-feasible arc $a = (i_t, j'_t)$ for which $p_a + p(j'_t)$ is maximum and ties are broken randomly.

**PILOT pathfinder** The PILOT pathfinder extends the greedy algorithm by incorporating a look-ahead mechanism as described in [7]. Instead of always extending the current partial solution with the arc $a = (i_t, j'_t)$ for which $p_a + p(j'_t)$ is maximum, for each possible extension a lower bound on the objective is computed (using the greedy algorithm described above). Then, the extension is performed which results in the highest value and ties are, again, broken randomly.

**Beam Search pathfinder** The beam search pathfinder is based on the beam search algorithm [13]. Beam search uses a set of partial solutions of size $k_{\text{bw}}$, called the beam, and evaluates the $k_{\text{ext}}$ most promising extensions to these. For this problem we choose the extensions based on the greedy criterion as explained before. In the end, all partial solutions of the beam are complete and the best one among those is taken as the final solution.

## 5 Computational Results

The developed algorithm is tested on a set of benchmark instances based on real-world data, which are described in Section 5.1. First, computational results on a set of instances are shown, which previously also have been approached by exact algorithms [5]. Then, in the second set of experiments, the developed pathfinder methods are compared on a different set of instances with stricter battery constraints to better highlight the differences of these methods.

### 5.1 Instance Description

The basis of all instances is the road network of Vienna, Austria obtained with OpenStreetMap data[1]. We assume potential locations for stations at supermarkets, parking lots, and subway stations. The number of slots for each

[1] `https://www.openstreetmap.org/`

station is chosen uniformly at random between 1 and 10. Station opening and slot costs are chosen uniformly at random from $F_i \in \{9000, \ldots, 64000\}$ and $Q_i \in \{22000, \ldots, 32000\}$ Euro, respectively. We further assume fast charging slots with a maximum charging rate per slot of 50 kW. The set of homogeneous cars is based on the data of the Smart ED car with an acquisition cost of 20000 Euro, a battery capacity of 17.6 kWh, and a maximum charging rate of 17.6 kW. The data for the requested trips is based on real Taxi data of one week in spring provided by a Taxi provider of Vienna. For each trip we are given the starting and ending point, and the duration and battery consumption is computed using the routing framework by Prandtstetter et al. [14]. Furthermore, we are given a starting and ending time, which is discretized in time intervals of 15 minutes, resulting in 672 time periods. The profit is computed by assuming a rate of 0.3 Euro / minute which approximately corresponds to the rate of local car-sharing systems. We assume that each trip can start and end at the three closest stations within a walking distance of 5 minutes. Therefore, the original road network $G$ is extended by introducing an arc $a' = (j, i)$ for each arc $a = (i, j) \in A$ of the original network and setting $l_{ji} = l_{ij}$. We assume a walking speed of 1.34 $m/s$ and compute a shortest path using Dijkstra's algorithm from the starting and the ending point of each request to all stations using the walking time as arc weight. The instance contains 693 potential station locations and 37965 trips.

As the size of the whole instance is very large, for the algorithm evaluation we filtered the instance and use only a subset of the trips and stations. Figure 2 shows four subsets of instances $I_1, \ldots, I_4$ based on the political districts of Vienna.

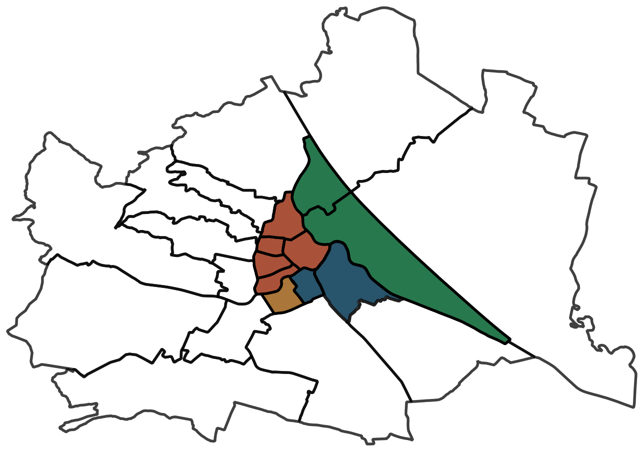

**Fig. 2.** The 23 districts of Vienna, Austria with the highlighted subset of districts which are used as instances.

The first instance $I_1$ uses only the potential stations and requests starting and ending in one of the five red districts. For $I_2$ the yellow district is added and for $I_3$ the yellow and the blue districts. Finally, the largest instance $I_4$ uses all ten colored districts. Furthermore, the instance have properties shown in Table 1, where $r_K$ is the average number of requests per hour, and $\overline{b_K}$ is the average energy consumption of the trips relative to the battery capacity of the Smart ED car in percent.

**Table 1.** Additional instance data.

| Instance | $\|S\|$ | $\|K\|$ | $r_K$ | $\overline{b_K}$ |
|---|---|---|---|---|
| $I_1$ | 105 | 108 | 0,68 | 3,10 |
| $I_2$ | 131 | 209 | 1,29 | 3,28 |
| $I_3$ | 198 | 719 | 4,31 | 3,37 |
| $I_4$ | 280 | 1347 | 8,04 | 3,27 |

Each such created instance is further split into specific instances by specifying a maximum budget and a maximum number of cars.

These instances are preprocessed in two ways:

1. Stations in which no requests can start or end are not considered.
2. The times of the requested trips are shifted forward in time so that the first fulfillable request starts at $t = 0$ and $t_{\max}$ is given by the ending time of the last fulfillable request. Thereby, the size of $T$ and hence the size of the TELN decreases which results in a faster solution evaluation.

## 5.2 Comparison to Exact Algorithms

First, we compare the results of the developed algorithm with exact methods based on integer linear programming (ILP) models developed by Brandstätter et al. [5]. They proposed several models and three of them (C1, C2, F1) turned out to performed best. Model C1 uses connectivity cuts and continuous battery tracking, C2 uses connectivity cuts and battery-infeasible path cuts, and model F1 is a multi-commodity flow formulation with continuous battery tracking. Table 2 shows the results of our proposed VNS with the greedy pathfinder compared to the results obtained from the exact models. We used instances with different budget constraints out of $W \in \{1000000, 2000000, 3000000, 4000000, 5000000\}$ Euro and the number of cars $H$ is restricted by either 10, 25, or 50. Here we used only the district subsets of $I_1$ and $I_2$ to be able to make comparisons. For instances $I_3$ and $I_4$ the state-of-the-art exact approaches are not able to produce reasonable results anymore. Table 2 shows the average objective value over 30 independent runs of the VNS ($\overline{\text{obj}}$), their associated standard deviation (sd), and the best objective value over these 30 runs ($\text{obj}^b$). Each run was performed on an Intel Xeon E5-2643 processor and terminated after 3600 seconds. The best solution, however, was usually found earlier (depending on the size of the instance) as the algorithm converged to its final solution. For the models the table

**Table 2.** Comparison to exact integer linear programming approaches

| Instance | | | VNS | | | C1 | | | C2 | | | F1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $W$ | $H$ | $I$ | $\overline{\text{obj}}$ | sd | obj$^b$ | obj$^{lb}$ | obj$^{ub}$ | gap | obj$^{lb}$ | obj$^{ub}$ | gap | obj$^{lb}$ | obj$^{ub}$ | gap |
| 1M | 10 | $I_1$ | 13559,0 | 303,7 | 13769 | 14045 | 14045,0 | 0,0 | 14045 | 14046,3 | 0,0 | 14045 | 14045,0 | 0,0 |
| 2M | 10 | $I_1$ | 19563,2 | 489,6 | 20444 | 21899 | 21901,0 | 0,0 | 21899 | 21899,0 | 0,0 | 21899 | 21901,1 | 0,0 |
| 3M | 10 | $I_1$ | 21411,1 | 220,2 | 21557 | 21956 | 21956,0 | 0,0 | 21956 | 21956 | 0,0 | 21956 | 21956,0 | 0,0 |
| 4M | 10 | $I_1$ | 21430,4 | 198,1 | 21557 | 21956 | 21956,0 | 0,0 | 21956 | 21956,0 | 0,0 | 21956 | 21956,0 | 0,0 |
| 5M | 10 | $I_1$ | 21493,1 | 123,1 | 21557 | 21956 | 21956,0 | 0,0 | 21956 | 21956,0 | 0,0 | 21956 | 21956,0 | 0,0 |
| 1M | 25 | $I_1$ | 13542,8 | 338,5 | 13769 | 14045 | 14045,0 | 0,0 | 14045 | 14046,2 | 0,0 | 14045 | 14046,2 | 0,0 |
| 2M | 25 | $I_1$ | 23467,8 | 320,2 | 24017 | 18302 | 25279,0 | 38,1 | 25279 | 25281,2 | 0,0 | 25279 | 25281,4 | 0,0 |
| 3M | 25 | $I_1$ | 28089,8 | 538,9 | 29168 | 30845 | 30845,0 | 0,0 | 30845 | 30845,0 | 0,0 | 30845 | 30848,0 | 0,0 |
| 4M | 25 | $I_1$ | 29807,1 | 359,4 | 30441 | 31215 | 31215,0 | 0,0 | 31215 | 31215,0 | 0,0 | 31215 | 31215,0 | 0,0 |
| 5M | 25 | $I_1$ | 30410,1 | 190,7 | 30686 | 31215 | 31215,0 | 0,0 | 31215 | 31215,0 | 0,0 | 31215 | 31215,0 | 0,0 |
| 1M | 50 | $I_1$ | 13625,1 | 201,0 | 13769 | 14045 | 14045,0 | 0,0 | 14045 | 14045,0 | 0,0 | 14045 | 14299,0 | 1,8 |
| 2M | 50 | $I_1$ | 23511,4 | 342,7 | 24086 | 25147 | 25291,3 | 0,6 | 25279 | 25279,0 | 0,0 | 25186 | 25645,5 | 1,8 |
| 3M | 50 | $I_1$ | 28783,9 | 524,6 | 29858 | 26057 | 31140,2 | 19,5 | 26837 | 31135,5 | 16,0 | 31131 | 31134,1 | 0,0 |
| 4M | 50 | $I_1$ | 31545,2 | 410,6 | 32403 | 30948 | 33971,0 | 9,8 | 33708 | 33952,8 | 0,7 | 33877 | 34197,0 | 0,9 |
| 5M | 50 | $I_1$ | 33154,4 | 425,9 | 33754 | 34093 | 34197,0 | 0,3 | 34197 | 34197,0 | 0,0 | 34197 | 34197,0 | 0,0 |
| 1M | 10 | $I_2$ | 23142,6 | 389,6 | 23515 | 24403 | 24403,0 | 0,0 | 24403 | 24403,0 | 0,0 | 24403 | 24403,0 | 0,0 |
| 2M | 10 | $I_2$ | 32032,7 | 748,9 | 33351 | 36916 | 37277,0 | 1,0 | 36846 | 37277,0 | 1,2 | 37277 | 37280,5 | 0,0 |
| 3M | 10 | $I_2$ | 37041,1 | 873,7 | 38431 | 40822 | 40822,0 | 0,0 | 40822 | 40822,0 | 0,0 | 40822 | 40822,0 | 0,0 |
| 4M | 10 | $I_2$ | 38967,6 | 347,9 | 39459 | 40822 | 40822,0 | 0,0 | 40822 | 40822,0 | 0,0 | 40822 | 40822,0 | 0,0 |
| 5M | 10 | $I_2$ | 38967,2 | 320,1 | 39496 | 40822 | 40822,0 | 0,0 | 40822 | 40822,0 | 0,0 | 40822 | 40822,0 | 0,0 |
| 1M | 25 | $I_2$ | 23141,2 | 436,7 | 23542 | 24403 | 24403,0 | 0,0 | 24403 | 24403,0 | 0,0 | 22675 | 27100,2 | 19,5 |
| 2M | 25 | $I_2$ | 37457,9 | 818,4 | 38415 | 27340 | 40952,2 | 49,8 | 24809 | 40957,0 | 65,1 | 39667 | 43608,0 | 9,9 |
| 3M | 25 | $I_2$ | 46500,5 | 1003,7 | 48301 | 36783 | 53172,5 | 44,6 | 52506 | 53166,0 | 1,3 | 52392 | 54470,6 | 4,0 |
| 4M | 25 | $I_2$ | 49988,0 | 1092,5 | 51987 | 58219 | 58253,0 | 0,1 | 58160 | 58253,0 | 0,2 | 58155 | 58253,0 | 0,2 |
| 5M | 25 | $I_2$ | 53639,6 | 981,8 | 55736 | 58253 | 58253,0 | 0,0 | 58253 | 58253,0 | 0,0 | 58253 | 58253,0 | 0,0 |
| 1M | 50 | $I_2$ | 22962,8 | 591,7 | 23542 | 24403 | 24403,0 | 0,0 | 22895 | 24430,1 | 6,7 | 23180 | 27745,0 | 19,7 |
| 2M | 50 | $I_2$ | 37349,1 | 825,6 | 38536 | 24852 | 40985,6 | 64,9 | 24031 | 40987,0 | 70,6 | 38940 | 43853,8 | 12,6 |
| 3M | 50 | $I_2$ | 47953,0 | 837,4 | 49381 | 36783 | 53564,4 | 45,6 | 36783 | 53502,5 | 45,5 | 49906 | 55732,9 | 11,7 |
| 4M | 50 | $I_2$ | 54131,0 | 1266,0 | 57060 | 47877 | 62012,2 | 29,5 | 47877 | 62014,9 | 29,5 | 61343 | 62958,9 | 2,6 |
| 5M | 50 | $I_2$ | 59358,4 | 1453,9 | 61943 | 57238 | 66494,4 | 16,2 | 57238 | 66494,4 | 16,2 | 65783 | 66971,0 | 1,8 |

shows the lower bound ($\text{obj}^{lb}$), the upper bound ($\text{obj}^{ub}$), and the optimality gap (gap). The models are solved using CPLEX 12.6.3 with a time limit of 6 hours.

The results in Table 2 show that the VNS is barely able to reach the results ($\text{obj}^{lb}$) of the exact methods when the gap is below 20%. For the larger instances $i_1$ with $H = 50$, however, the VNS starts finding better results than model C1 and C2. The average gap of the VNS to the optimal value of the optimally solved instances is about 6.1%. After an analysis of the results we observed that the main reason for the worse performance on some instances lies in the inaccuracy of the solution evaluation. Although the greedy pathfinder is often able to find the optimal path through the TELN, the overall evaluation procedure is just an approximation since the path of each car is evaluated separately in succession. Therefore, even if the VNS would generate an optimal solution, the pathfinder heuristic would not be able to identify it as such because it might be assigned a worse objective value from the procedure which calculates the accepted trips.

## 5.3 Pathfinder Results for Larger Instances

In the second set of experiments the impact of the used pathfinder (greedy (G), PILOT (P), and beam search (B)) on the final results is investigated in more detail. The instances, as described in Section 5.1 have a low trip density and average energy consumption per trip. Preliminary results showed that for these instances the greedy pathfinder always performed best because it is the fastest algorithm and the PILOT and beam search method were never able to find better paths. Therefore, we now assume an adapted demand forecast in which the system is more utilized and the requested trips are longer. So, we change the instances in the following way:

- The starting times of all requests of the instance are scaled in such a way that they all start within an 8 hour period.
- The battery consumption of each request is increased by a factor of 6.

By applying these adaptions the instance becomes, on the one hand, more *crowded* so that more trips are requested in a short time period and, on the other hand, it becomes more battery restricted. Therefore, the used pathfinder is more crucial to the solution evaluation. The results of the different algorithms are shown in Table 3 and, again, 30 runs per instance are performed with a maximum run-time of 3600 seconds.

In this table the instances are grouped by the subset of districts $i_1, \ldots, i_3$ and the maximum number of cars $H \in \{10, 25, 50\}$ resulting in 5 instances with different budget constraints per row. For each row the geometric mean ($\overline{\text{obj}_\text{g}}$), the number of instances for which the algorithm yields the best average objective value over all 30 runs (#best), and the results of the statistical tests are given. For testing the statistical differences between the algorithms we performed one-sided Wilcoxon rank sum tests using an error level of 5%. The entries of the table corresponds to the number of instances for which algorithm A yields statistically significant better results than algorithm B, i.e., the entries in column $P$ below

**Table 3.** Results of the different pathfinders for the altered instances aggregated by used subset of districts $i$ and number of available cars $H$.

| Instance | | $\overline{\mathrm{obj}_\mathrm{g}}$ | | | #best | | | #G$>_{sig}$ | | #P$>_{sig}$ | | #B$>_{sig}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | H | G | P | B | G | P | B | P | B | G | B | G | P |
| $I_1$ | 10 | 17800,5 | 18040,9 | 17965,7 | 1 | 3 | 1 | 1 | 1 | 4 | 4 | 3 | 0 |
| | 25 | 23752,6 | 23933,8 | 23886,5 | 1 | 4 | 0 | 1 | 1 | 4 | 0 | 4 | 0 |
| | 50 | 25332,2 | 25229,1 | 25102,8 | 3 | 2 | 0 | 3 | 3 | 2 | 3 | 0 | 0 |
| $I_2$ | 10 | 28529,0 | 28829,0 | 28684,0 | 0 | 5 | 0 | 0 | 1 | 1 | 2 | 0 | 0 |
| | 25 | 39973,8 | 39680,7 | 39220,2 | 5 | 0 | 0 | 3 | 5 | 0 | 2 | 0 | 0 |
| | 50 | 42788,8 | 42317,0 | 42019,2 | 5 | 0 | 0 | 4 | 5 | 0 | 1 | 0 | 0 |
| $I_3$ | 10 | 46528,5 | 47068,0 | 46257,9 | 1 | 4 | 0 | 0 | 2 | 2 | 4 | 0 | 0 |
| | 25 | 71937,7 | 71052,8 | 69499,3 | 4 | 1 | 0 | 3 | 4 | 1 | 4 | 0 | 0 |
| | 50 | 81131,1 | 78513,6 | 76170,1 | 4 | 1 | 0 | 4 | 4 | 1 | 4 | 0 | 0 |
| $I_4$ | 10 | 46648,1 | 46499,7 | 45775,3 | 3 | 2 | 0 | 1 | 3 | 1 | 3 | 0 | 0 |
| | 25 | 71833,6 | 69268,5 | 66719,6 | 5 | 0 | 0 | 4 | 5 | 0 | 3 | 0 | 0 |
| | 50 | 81547,6 | 77482,9 | 75111,6 | 5 | 0 | 0 | 4 | 5 | 0 | 3 | 0 | 0 |

#G$>_{sig}$ are the number of instances the greedy pathfinder finds better results than the PILOT pathfinder.

Table 3 shows that for the smallest instance $I_1$ and for the instances with $H = 10$ both the PILOT and the beam search pathfinder tend to yield better results than the greedy pathfinder. As the instance size or the number of available cars increases, however, the greedy pathfinder starts performing better than the other two. Especially the beam search pathfinder is outperformed by the greedy and PILOT pathfinder, as it could not find significantly better results than the latter for any of our test instances. This behavior can be explained by the higher run-time complexity for the PILOT and beam search pathfinder as the instance size grows. When the greedy pathfinder is used, the algorithm can perform more iterations overall and is therefore able to find better solutions.

## 6 Conclusions and Future Work

In this work a heuristic algorithm for finding and designing charging stations for electric vehicles is presented. It is based on a bi-level model formulation in which the first level decides on the station locations, number of slots per station, and the total number of cars by using a variable neighborhood search. Then, for each generated solution candidate of the first level, a path-based heuristic is used for the evaluation, i.e., deciding which trips can be accepted by the system. For the path-based heuristic three different path-finders are implemented and compared to each other. The results show that for smaller instances and instances with a smaller number of available cars that have a high trip density and many trips with a high energy consumption, the PILOT pathfinder performs best in most of these instances. On the other hand, if the instance size is getting larger, the trips are getting shorter, or the trip density decreases, the greedy pathfinder outperformed the other two. The beam search pathfinder was, however, not able to consistently find any significantly better results than the greedy or the PILOT method. The implemented algorithm was further compared with several exact methods based on integer linear programming from the literature. The results show that for those instances which are hard to solve by exact approaches and thus have high optimality gaps, the VNS is able to yield better results. However, it was not able to find solutions with the same quality for those instances that could be solved to optimality. The main reason for this is that the solution evaluation using the path-based heuristic is not exact and therefore the algorithm is not able to find the optimal trips to be accepted. An exact evaluation might improve these results but the algorithm would not scale well with the instance size anymore.

Although we considered many aspects of a real-world electric car-sharing system, some operational decisions are neglected yet. As one-way car-sharing systems tend to fall out of balance over time because of the not uniformly distributed demand in the operational area some kind of re-distribution of the vehicles must be performed. These re-distributions can be user-based, e.g., by giving the users incentives, or system-based. When using the latter, the re-distribution is usually performed by dedicated relocators who have to move cars between stations. This step is important for the operational decisions of such an electric car-sharing system but can also be considered for the strategic decisions of the station planning.

Another interesting direction for future work are free-floating systems, in which the users can rent and return the cars anywhere within the operational area. When using electric cars in such a system, charging stations still have to be planned and the re-distribution may become even more important. The impact of using such a system on the trip acceptance rate, the user experience, and on the strategic planning is promising and relevant for future research.

## References

1. Asamer, J., Reinthaler, M., Ruthmair, M., Straub, M., Puchinger, J.: Optimizing charging station locations for urban taxi providers. Transportation Research Part A: Policy and Practice 85, 233–246 (2016)
2. Boyacı, B., Zografos, K.G., Geroliminis, N.: An optimization framework for the development of efficient one-way car-sharing systems. European Journal of Operational Research 240(3), 718–733 (2015)
3. Brandstätter, G., Gambella, C., Leitner, M., Malaguti, E., Masini, F., Puchinger, J., Ruthmair, M., Vigo, D.: Overview of optimization problems in electric car-sharing system design and management. In: Dynamic Perspectives on Managerial Decision Making, pp. 441–471. Springer (2016)
4. Brandstätter, G., Kahr, M., Leitner, M.: Determining optimal locations for charging stations of electric car-sharing under stochastic demand (2016), submitted
5. Brandstätter, G., Leitner, M., Ljubić, I.: Location of charging stations in electric car sharing systems (2016), submitted

6. Cavadas, J., de Almeida Correia, G.H., Gouveia, J.: 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 (2015)
7. Duin, C., Voß, S., et al.: The pilot method: A strategy for heuristic repetition with application to the steiner problem in graphs. Networks 34(3), 181–191 (1999)
8. Frade, I., Ribeiro, A., Gonçalves, G., Antunes, A.: 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 (2011)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
10. Ge, S., Feng, L., Liu, H.: The planning of electric vehicle charging station based on grid partition method. In: Electrical and Control Engineering (ICECE), 2011 International Conference on. pp. 2726–2730. IEEE (2011)
11. Hess, A., Malandrino, F., Reinhardt, M.B., Casetti, C., Hummel, K.A., Barceló-Ordinas, J.M.: Optimal deployment of charging stations for electric vehicular networks. In: Proceedings of the first workshop on Urban networking. pp. 1–6. ACM (2012)
12. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research 24(11), 1097–1100 (1997)
13. Ow, P.S., Morton, T.E.: Filtered beam search in scheduling. The International Journal Of Production Research 26(1), 35–62 (1988)
14. Prandtstetter, M., Straub, M., Puchinger, J.: On the way to a multi-modal energy-efficient route. In: Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE. pp. 4779–4784. IEEE (2013)
15. Weikl, S., Bogenberger, K.: Relocation strategies and algorithms for free-floating car sharing systems. IEEE Intelligent Transportation Systems Magazine 5(4), 100–111 (2013)