# Distributing Battery Swapping Stations for Electric Scooters in an Urban Area*

Thomas Jatschka[1], Fabio F. Oberweger[1], Tobias Rodemann[2], and Günther R. Raidl[1]

[1] Institute of Logic and Computation, TU Wien, Austria
{tjatschk,raidl}@ac.tuwien.ac.at, e1551139@student.tuwien.ac.at
[2] Honda Research Institute Europe, Germany
tobias.rodemann@honda-ri.de

**Abstract.** We investigate the problem of setting up battery swapping stations for electric scooters in an urban area from a computational optimization point of view. For the considered electric scooters batteries can be swapped quickly in a few simple steps. Depleted batteries are recharged at these swapping stations and provided again to customers once fully charged. Our goal is to identify optimal battery swapping station locations as well as to determine their capacities appropriately in order to cover a specified level of assumed demand at minimum cost. We propose a Mixed Integer Linear Programming (MILP) formulation that models the customer demand over time in a discretized fashion and also considers battery charging times. Moreover, we propose a Large Neighborhood Search (LNS) heuristic for addressing larger problem instances for which the MILP model cannot practically be solved anymore. Prototype implementations are experimentally evaluated on artificial benchmark scenarios. Moreover, we also consider an instance derived from real-world taxi and bus stop shelter data of Manhattan. With the MILP model, instances with up to 1000 potential station locations and up to 2000 origin/destination demand pairs can be solved to near optimality, while for larger instances the LNS is a highly promising choice.

**Keywords:** Facility location problem · e-mobility · battery swapping stations · mixed integer linear programming · large neighborhood search

## 1 Introduction

Recharging the batteries of electric vehicles is usually a time-consuming process that hinders the large-scale adoption of such vehicles, especially when their range without reloading is too limited. An alternative possibility is to build electric vehicles in which the batteries can be replaced with charged ones. Batteries for electric scooters are compact enough to be replaced directly by any customer in a few simple steps. Replacement batteries are provided in exchange for the used

* Thomas Jatschka acknowledges the financial support from Honda Research Institute Europe. We thank Honda R&D Co., Ltd. for technical insights.

ones at swapping stations. Returned batteries are recharged at these stations, and once fully charged, they are again provided for exchange.

We aim at investigating how to best distribute such battery swapping stations in a given urban area and how many battery slots and corresponding batteries are required at each station. Our optimization goal is to minimize the setup costs for stations in dependence of their numbers of slots and required batteries in order to cover a specified amount of user demand over multiple consecutive time periods. It is assumed that customers who want to change batteries specify their trip data (origin, destination, approximate time) online and are automatically assigned to an appropriate station for the exchange (if one exists). This way, a better utilization of the swapping stations can be achieved. However, such an automated assignment also needs to consider a certain customer dropout as not every customer is willing to travel to a predestined station if the detour is long. We assume that all scooters in our system are homogeneous and therefore require the same batteries and have the same range. Moreover, since the scooters are operating in an urban area, it is safe to assume that a scooter's range is usually larger than the length of a customer's single trip. Hence, we do not consider multiple battery swapping stops for a single trip. In fact, a scooter battery is typically exchanged after multiple trips only. We model this problem as a mixed integer linear program (MILP). Smaller problem instances can be solved by directly applying a state-of-the-art MILP solver. To address the aspect of scalability to larger instances, where the MILP solver does not yield satisfactory solutions anymore, a *Large Neighborhood Search* (LNS) heuristic is proposed. The approaches are experimentally evaluated on artificial benchmark scenarios as well as one instance derived from real-world yellow taxi trip data and bus stop shelter station data of Manhattan.

Section 2 reviews relevant related work. Section 3 presents the problem formalization in the form of a MILP. The LNS heuristic is described in Section 4. Section 5 explains how the benchmark scenarios are generated. Experimental results of the proposed solution methods are given in Section 6. Finally, Section 7 concludes this article and gives an outlook on future work.

## 2 Related Work

In general, our problem can be classified as a location-allocation optimization problem [1]. Specifically, our problem is closely related to the capacitated multiple allocation Fixed Charge Facility Location Problem (FLP) [2] in which customers need to be assigned to facilities in order to satisfy their demand while minimizing costs for building facilities and serving customers. Moreover, the customer demand can be split arbitrarily between multiple facilities. When allocating customers to facilities from the perspective of the facility provider without considering the customers' preferences, one frequently has to expect a certain amount of customer dropout which we model with the help of a decay function as done in, e.g., [3–5]. Facility location problems with time dependent parameters

are also referred to as multi-period FLPs [2]. One example for a multi-period FLP can be found in [6], where the dynamic maximal covering problem is considered.

Moreover, our problem exhibits similarities with the Capacitated Deviation-Flow Refueling Location Model (CDFRLM) introduced in [7], which is an extension of the Flow Refueling Location Model (FRLM) introduced by Kuby and Lim [8]. The FRLM aims to locate a fixed amount of refueling stations to maximize the total flow volume refueled. Several extensions of the FRLM have been proposed in the last years, such as the capacitated FRLM [9] in which the demand a station can satisfy is limited. The Deviation Flow Refueling Location Model (DFRLM) [5] relaxes the FRLM by allowing customers to deviate from their shortest O/D pair paths in order to go to a refueling station. Moreover, it is assumed that the number of customers willing to take a deviation from the shortest path is exponentially decreasing with the length of the deviation. In [7], the Capacitated Deviation-Flow Refueling Location Model (CDFRLM) is presented which also introduces station capacities to the DFRLM,

While there already exists work for setting up a system of battery swapping stations, e.g., [10], [11], to the best of our knowledge, there is no previous work that considers specifically the aspect of recharging and reusing returned batteries and its implications concerning station capacities when optimizing station locations and configurations.

## 3 The Multi-Period Battery Swapping Station Location Problem

In this section we formalize the problem of setting up battery swapping stations for electric scooters in an urban area. The *Multi-Period Battery Swapping Station Location Problem* (MBSSLP), as we call it, minimizes the costs for setting up battery swapping stations to satisfy a requested expected total demand over a whole day. To be able to consider battery charging times, we consider a day in a discretized fashion as a set of equally long consecutive time intervals given as a set of the start times $\mathcal{T}$ of the intervals; w.l.o.g., we assume $\mathcal{T} = \{1, \ldots, t_{\max}\}$. We make the simplifying assumption that charging any battery always takes the same time and only completely recharged batteries are provided to customers again. Moreover, as trips in an urban environment are usually rather short, we further assume that trips start and end in the same time interval.

Let $G = (V, A, w)$ be a weighted directed graph with node set $V$ corresponding to all relevant geographic locations, arc set $A \subseteq V \times V$, corresponding to shortest paths between locations, and arc weights $w : A \to \mathbb{R}^+$ representing the respective travel times. We assume battery swapping stations can be set up at a subset of locations $L = \{1, \ldots, n\} \subseteq V$. Moreover, each location $l \in L$ has associated a maximal number of possible battery charging slots $s_l \geq 0$, fixed setup cost $c_l$ for setting up a station at this location, and building costs per slot $c_l^s \geq 0$. Customer travel demands are given by origin-destination (O/D) pairs $Q \subseteq V \times V$; let $m = |Q|$. The expected number of users that need to change batteries on trip $q \in Q$ during a time interval $t \in \mathcal{T}$ is denoted as $d_q^t$. The

minimal amount of expected total customer demand that shall be satisfied over all time intervals in $\mathcal{T}$ is denoted by $d_{\min}$. Moreover, we are given a maximum detour length $w_{\max}^{\text{detour}}$ by which a feasible path including a battery swap for some $q \in Q$ may be longer than a shortest path from the origin to the destination of $q$. Finally, the number of time intervals required for completely recharging a battery is referred to as $t^{\text{c}}$.

It is assumed that customers would always take a shortest possible path $p_q$ for an O/D pair $q = (u, v) \in Q$, except when they have to make a detour for swapping batteries. Let the set of arcs of a shortest path $p_{uv}$ from node $u \in V$ to node $v \in V$ be $A(p_{uv}) \subseteq A$ and its length $w(p_{uv}) = \sum_{e \in A(p_{uv})} w(e)$. Moreover, we consider for an O/D pair $q = (uv) \in Q$ a shortest path that includes a certain location $l \in L$ as intermediate stop and denote it by $p_q^l$. The combination of a shortest path from $u$ to $l$ and a shortest path from $l$ to $v$ forms such a shortest path $p_q^l$, and its length is $w(p_q^l) = w(p_{ul}) + w(p_{lv})$. Let $L_q$ be the set of locations $l \in L$ for which $w(p_q^l) \leq w(p_q) + w_{\max}^{\text{detour}}$ for $q \in Q$, i.e., the locations that may be used for battery swaps for O/D pair $q$.

A solution to the MBSSLP is primarily given by a pair of vectors $x = (x_l)_{l \in L} \in \{0,1\}^n$ and $y = (y_l)_{l \in L}$ with $y_l \in \{0, \ldots, s_l\}$, where $x_l = 1$ indicates that a swapping station is to be established at location $l$ and $y_l$ is the respective number of battery slots. Moreover, let $a_{ql}^t$ denote the part of the expected demand of O/D pair $q \in Q$ which we assign to a location $l \in L_q$ during time period $t \in \mathcal{T}$. Similarly to [5], we consider the loss of users in dependence of the detour length by applying a penalty coefficient $g(q, l)$ to $a_{ql}^t$ in order to obtain the actually expected satisfied demand $\tilde{a}_{ql}^t$ of O/D pair $q$ at location $l$. As suggested in [12, 5] we use the sigmoid function for this penalty coefficient, i.e., $g(q, l) = 1/(1 + \alpha e^{\beta(w(p_q^l) - w(p_q)) - \delta_q})$, where $w(p_q^l) - w(p_q)$ is the detour distance for going to the swapping station, $\delta_q$ is a reference distance, and $\alpha$ and $\beta$ are parameters determine the shape of the function.

Based on the variables $x, y, a$, and $\tilde{a}$ the MBSSLP can be expressed as the following MILP.

$$\min \sum_{l \in L} (c_l x_l + c_l^{\text{s}} y_l) \tag{1}$$

$$x_l \cdot s_l \geq y_l \qquad \forall l \in L \tag{2}$$

$$\tilde{a}_{ql}^t = g(q, l) \cdot a_{ql}^t \qquad \forall t \in \mathcal{T},\ q \in Q,\ l \in L_q \tag{3}$$

$$\sum_{l \in L_q} a_{ql}^t \leq d_q^t \qquad \forall t \in \mathcal{T},\ q \in Q \tag{4}$$

$$\sum_{t' = \max(1, t - t^{\text{c}})}^{t} \sum_{q \in Q | l \in L_q} \tilde{a}_{ql}^{t'} \leq y_l \qquad \forall t \in \mathcal{T},\ l \in L \tag{5}$$

$$\sum_{t=1}^{t_{\max}} \sum_{q \in Q} \sum_{l \in L_q} \tilde{a}_{ql}^t \geq d_{\min} \tag{6}$$

$$x_l \in \{0,1\} \qquad \forall l \in L \qquad (7)$$

$$y_l \in \{0,\dots,s_l\} \qquad \forall l \in L \qquad (8)$$

$$0 \le a^t_{ql}, \tilde{a}^t_{ql} \le s_l \qquad \forall t \in \mathcal{T},\ q \in Q,\ l \in L_q \qquad (9)$$

The goal of the objective function (1) is to find a feasible solution that minimizes the setup costs for stations and their battery slots. Inequalities (2) ensure that battery slots can only be allocated to a location $l \in L$ if a station is opened there. For better readability equalities (3) introduce variables $\tilde{a}^t_{ql}$ by applying the penalty coefficients $g(q,l)$ to variables $a^t_{ql}$. Constraints (4) enforce that the total demand assigned from an O/D pair $q$ to locations does not exceed $d^t_q$ for all $t \in \mathcal{T}$. Inequalities (5) ensure the required capacity $y_l$ at all locations over all time intervals. Note that by using $\tilde{a}^t_{ql}$ instead of $a^t_{ql}$ in (5), we "overbook" stations to consider the expected case, similarly as in [13]. Inequalities (5) also model that swapped batteries can be reused after $t^{\mathrm{c}}$ time intervals. The minimal satisfied demand to be fulfilled over all time intervals is expressed by inequality (6). Finally, the domains of the variables are given in (7)–(9).

## 4 Large Neighborhood Search

Large Neighborhood Search (LNS) [14] is a prominent metaheuristic for addressing difficult combinatorial optimization problems, which builds upon effective lower-level heuristics. A basic LNS in essence follows a classical local search framework, but usually much larger neighborhoods are considered in each iteration. The key-idea is to search these neighborhoods not in a naive enumerative way but to apply some "more clever" problem-specific procedure to solve the subproblem induced by each neighborhood in order to obtain the best or a promising heuristic solution from the neighborhood. Frequently, LNS follows a destroy and recreate scheme: A current incumbent solution is partially destroyed, typically by freeing a subset of the decision variables and fixing the others to their current values, and then repaired again by finding best or at least promising values for the freed variables.

We first show how to construct an initial solution in a fast greedy way. Afterwards, the search and destroy operators of our LNS are described.

### 4.1 Greedy Construction Heuristic

The construction heuristic generates a solution station-wise. In each iteration of the algorithm a new station is opened and demand is allocated to it. In order to decide at which location to open a station next, we first calculate how much additional demand a new station at each so far unused location could satisfy w.r.t. the already opened stations. The location with the highest ratio of additionally satisfied demand to corresponding building costs is then chosen for opening the next station.

To calculate the amount of demand a station $l \in L$ can satisfy, demand is assigned from each $q \in Q \mid l \in L_q$ for all time periods $t \in T$ to $l$ until either the

station's maximum capacity is exhausted or all demand has been assigned. The iteration order of $Q$ is hereby decided by the decay function $g$ such that O/D pairs with lower decay value w.r.t. $l$ are considered first.

The construction algorithm terminates when one of the following conditions is met: at least $d_{\min}$ demand is satisfied, stations are opened at all possible locations, or no more demand can be assigned to a station anymore.

### 4.2 Destroy and Repair Operators

Let $(x, y, a)$ be a solution to the MBSSLP. Moreover, let $L(x) \subseteq L$ be the set of locations for which $x_l = 1$. In a first step we create an undirected graph $G^L = (V, E)$ where $(u, v) \in E$ for $u, v \in V$ if and only if $\{u, v\} \subseteq L_q$ for at least one O/D pair $q \in Q$.

We then derive a set of locations $L_{\text{repair}}$ that are considered for repairing via an $(r, k)$-*repair operator*. The operator iteratively adds $k$ random node sets to $L_{\text{repair}}$ where each node set is generated by choosing a random vertex $v \in V$ as well as $r$ random neighbors of $v$ in $G^L$ (less if the degree of $v$ is less than $r$). Afterwards, $k$ random locations from $L(x)$ are added to $L_{\text{repair}}$. Should, during the generation of $L_{\text{repair}}$, a randomly selected vertex already be in $L_{\text{repair}}$ the repair operator chooses a new random vertex if possible. From $L_{\text{repair}}$ we derive the set $L_{\text{destroy}} = L_{\text{repair}} \cap L(x)$, and close all stations at these locations.

When repairing the solution, one needs to consider how much more demand needs to be satisfied in order to make the solution feasible again and how much demand from which O/D pairs is still available to be assigned to a station. For this purpose, let $D' = (d'^t_q)_{t \in T, q \in Q}$ be the demand not yet assigned to any opened location in the destroyed solution, i.e., $d'^t_q = d^t_q - \sum_{l \in L(x) \setminus L_{\text{destroy}}} a^t_{ql}$. Moreover, let $d_{\text{sat}}$ be the amount of total demand satisfied in the partially destroyed solution, i.e., $d_{\text{sat}} = \sum_{l \in L(x) \setminus L_{\text{destroy}}} \sum_{t=1}^{t_{\max}} \sum_{q \in Q} \tilde{a}^t_{ql}$. Hence, the goal of the repair function is to assign at least $d'_{\min} = d_{\min} - d_{\text{sat}}$ demand from $D'$ to the locations $L' = L_{\text{destroy}} \cup L_{\text{repair}}$. For this purpose, let $I(L', D', d'_{\min})$ be the residual MBSSLP instance in which $L$, $D = (d^t_q)_{t \in T, q \in Q}$, and $d_{\min}$ are replaced with $L'$,$D'$, and $d'_{\min}$. We determine a promising heuristic solution to $I(L', D', d'_{\min})$ using a relaxation of the MILP (1)–(9): Allowing the $y_l$ variables to be continuous, i.e., replacing (8) by $0 \leq y_l \leq s_l$, $\forall l \in L$, while still keeping the $x_l$ variables integral significantly speeds up the solving of the MILP. Obtained fractional values for $y_l$ are finally rounded up to obtain a feasible solution to the original MBSSLP again, assuming one exists.

Note that the described solving of the relaxation of the MILP followed by rounding can also be used as a standalone heuristic for the original MBSSLP, which is applicable as long as the instance is not too large. We refer to this approach as *y-Relaxed MILP Heuristic* ($\text{RMH}_y$). Additionally, we also considered solving the full linear relaxation of the original MILP, i.e., the linear program in which all $x_l$ as well as $y_l$ variables are continuous, and rounding up obtained fractional $x_l$ as well as $y_l$ values to the next integers; we call this heuristic *Linear Programming Heuristic* (LPH). In Section 6 we compare these approaches to each

other, showing that the $\text{RMH}_y$ heuristic is a better choice for repairing solutions than the LPH heuristic.

## 5 Test Instances

As no real problem instances are available to us we created artificial test instances with characteristics that might be expected in real scenarios. The creation of this instances is described next. Moreover, we derived one problem instance from real-world taxi trip and bus stop data of Manhattan as described in Section 5.2.

### 5.1 Random Instances for the MBSSLP

The instances are simplified scenarios modeled after a typical work day where people go to work in the morning and return home in the evening. Battery swapping stations as well as origin and destination locations of customers are located within a square of length $\lceil \xi\sqrt{n} \rceil$ with $\xi = 800$. We generate a network graph $G = (V, E)$ following a similar procedure as used in [7, 15] by first sampling $|V| = 5n$ random points from the square and then constructing an euclidean spanning tree w.r.t. $V$. Afterwards, $n$ additional randomly chosen edges $(u, v) \in V \times V$ are added to $E$.

The set of potential battery swapping station locations $L$ is generated by choosing $n$ random nodes from $V$. Costs for building a station are chosen uniformly at random from $\{50, \ldots, 70\}$ for each station. Costs for adding a battery slot to a station are set to 40. Each battery swapping station can have at most 70 battery slots.

Origin and destination locations are chosen from a random subset $V' \subseteq V$ with $|V'| = \min(\frac{m}{2}, 5n)$. To each $v \in V'$ a random weight $\gamma_v$ is assigned according to a log-normal distribution with mean $\mu = \ln(100)$ and standard deviation $\sigma = 0.5$. Moreover, we also assign weights $\gamma_q$ to each OD-pair $q = (u, v) \in V' \times V'$ such that $\gamma_q$ corresponds to $f_{\text{PDF}}(w(p_q), \mu, \sigma)$ with $f_{\text{PDF}}$ being the probability density function of a lognormal distribution with mean $\mu = \ln(5000)$ and standard deviation $\sigma = 0.2$. The total demand $d_q^{\text{total}}$ of an O/D-pair $q = (u, v)$ is then calculated as $d_q^{\text{total}} = \gamma_u \cdot \gamma_v \cdot \gamma_q$. We then set $Q$ to be the set of $m$ O/D-pairs $q$ of $V' \times V'$ for which $d_q^{\text{total}}$ is highest.

The swapping demand of each O/D-pair is distributed over 24 time periods, $\mathcal{T} = \{1, \ldots, 24\}$ and recharging a battery requires one time period, i.e., $t^c = 1$. We assume each customer to travel twice on his corresponding path, once in the morning to get to work and once in the evening to travel back home, and we assume that customers need to swap batteries once per trip counted here as demand. The demand during each time period $t \in \mathcal{T}$ is determined by two normal distributions $\mathcal{N}_{\text{morning}}(8, 1)$ and $\mathcal{N}_{\text{evening}}(18, 2)$, respectively. From each distribution 100 samples $t$ are generated and transformed to valid integral values by $t := (\lceil t \rceil \mod t_{\max}) + 1$. Afterwards, $d_q^{\text{total}}$ is distributed over $\mathcal{T}$ according to the frequency in which the time periods $t \in \mathcal{T}$ appear in the generated samples.

The maximal deviation distance of the users, $w_{\max}^{\text{detour}}$, is set to $\xi/2$ and the parameters of the distance decay function are set to $\alpha = 100$, $\beta = 0.1$, and $\delta_q = w_{\max}^{\text{detour}}/10$ for all $q \in Q$. Figure 1 shows the decay value $g(q, l)$ in dependence of the deviation distance $w(p_q^l) - w(p_q)$ with the chosen parameterization.

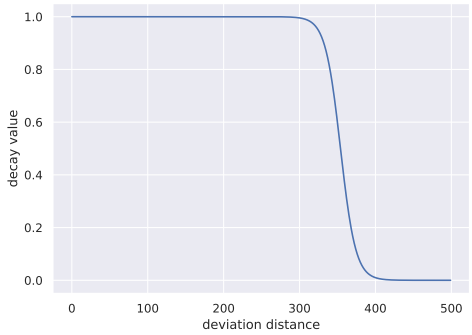



Fig. 1: Decay $g(q, l)$ in dependence of the deviation distance $w(p_q^l) - w(p_q)$.

Eight groups of test instances for different combinations of $n$ and $m$ have been generated as described in Section 5, and each group consists of thirty instances. In Section 6 we evaluate the instances with $d_{\min}$ being set either to 30% or to 80% of the total swapping demand.

### 5.2 Manhattan Instance

Next to artificial benchmark instances we also derived an instance from real-world yellow taxi trip data and bus stop shelter data of Manhattan, which we call here Manhattan instance. The underlying street network of the instance corresponds to the street network graph of Manhattan provided by the Python package OSMNX[3]. Origin/Destination pairs of our instance correspond to trips between the taxi zones[4] of Manhattan. The partitioning of Manhattan into taxi zones is shown in Figure 3. For each taxi zone one random origin and one random destination location were chosen from the set of nodes of the network graph that are associated with the corresponding taxi zone.

The set of O/D-pairs and their corresponding demands have been derived from the 2016 Yellow Taxi Trip Data[5]. The taxi data set was first preprocessed and all trips with invalid data as well as trips made on a weekend have been removed from the data set. Furthermore, we have also removed all trips which do not start and end in Manhattan. From the preprocessed data set we then

[3] https://github.com/gboeing/osmnx
[4] https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc
[5] https://data.cityofnewyork.us/Transportation/2016-Yellow-Taxi-Trip-Data/k67s-dv2t

extracted for each trip the pickup time, the pickup zone, the drop-off zone, as well as the passenger count. Each pickup time was rounded down to the nearest hour and afterwards an average daily passenger count for each triple (pickup hour, pickup zone, drop-off zone) was calculated. In total, the final table contains 4498 unique pickup/drop-off zone pairs which also constitute the instance's set of O/D pairs $Q$. These passenger counts correspond to the hourly demands $d_q^t$ of the O/D pairs $q \in Q$. Figure 2 shows on the left how the total demand over all O/D pairs is distributed over the time intervals. Figure 2 shows on the right how the lengths of the O/D pairs are distributed. Similarly to our benchmark instances, the trip lengths are approximately log-normal distributed with a mean between ln(5000) and $ln(6000)$. For the distance decay function and $w_{\max}^{\text{detour}}$ we

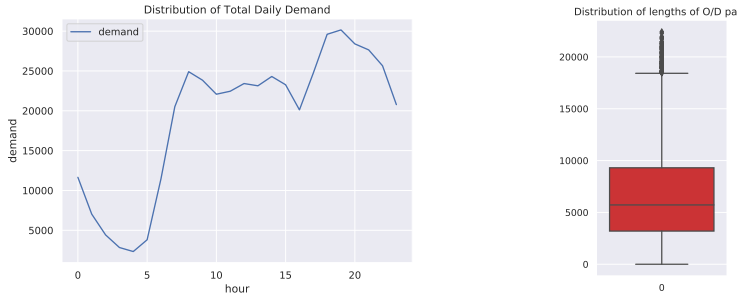



Fig. 2: Distributions of (a) demand and (b) trip length of the O/D pairs from the real-world data based instance.

use the same parameters as for the artificial benchmark instances.

The set of potential battery swapping station locations $L$ is derived from the bus stop shelters [6] of Manhattan by selecting 500 locations randomly. Figure 3 shows the distribution of the stations.

As shown in Figure 2 left the demand at each hour is quite high. Therefore we choose a capacity limit of 200 for each battery swapping station, The costs for building a station as well for adding a battery charging slot are chosen as for the artificial instances.

## 6 Computational Results

All algorithms were implemented in Julia[7] 1.4.2. All test runs have been executed on an Intel Xeon E5-2640 v4 2.40GHz machine in single-threaded mode with a time limit of thirty minutes. Gurobi[8] 8.1.0 was used for solving the MILPs.

First, we investigate the performance of the standalone MILP model given by Equations (1)–(9) as well as the standalone $\text{RMH}_y$ and the LPH approach.

[6] https://data.cityofnewyork.us/Transportation/Bus-Stop-Shelters/qafz-7myz
[7] https://julialang.org/
[8] https://www.gurobi.com/

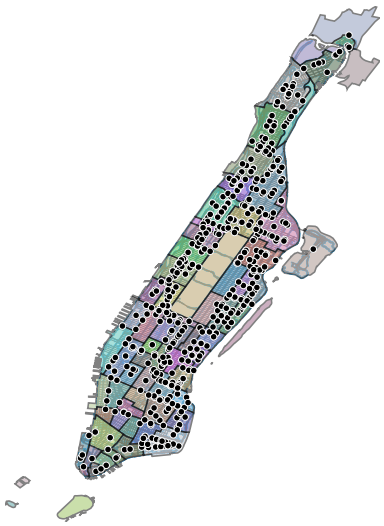


Fig. 3: Taxi zones of Manhattan and potential locations for swapping stations.

Afterwards, the results of the LNS are discussed. Finally, in Section 6.3 we present the results on the instance derived from real-world data for the LNS approach as well as the MILP models. All instances are evaluated with $d_{\min}$ being set either to 30% or to 80% of the total swapping demand. Hence, let $d_{\min}[\%]$ refer to $d_{\min}$ as percentage of the total swapping demand.

### 6.1 MILP Approaches

All MILP models were solved with Gurobi 8.1.0. In case no optimal solution was found within the time limit, the solver returned the best found feasible solution if it exists.

Table 1 shows a summary of the performance of the exact MILP approach, $\text{RMH}_y$ and LPH for each instance group in our benchmark set. Column "gap[%]" shows the average optimality gaps for each instance group, the median computation times are shown in column "time[s]", and column "$|L(x)|$" lists the average number of opened stations in the solutions. Note that the gaps listed for $\text{RMH}_y$ and LPH are determined also w.r.t. the lower bounds obtained by the original MILP.

Overall, with the exact MILP solving was aborted due to the time limit for almost all instances. However, for each instance at least one feasible solution was found. Instances with up to 1000 potential battery swapping stations and 2000 O/D-pairs can be solved by the MILP almost to optimality with a gap of less than 1%. For larger instances the optimality gaps deteriorate. Compared to the results of the original MILP model, $\text{RMH}_y$ yields in general better average optimality gaps for the three largest instance groups. The LPH approach was able to solve all instances to optimality w.r.t. the linear relaxation of the original MILP in

Table 1: Results of the original MILP, the $\text{RMH}_y$ heuristic, and the LPH heuristic.

(a) MILP results for $d_{\min}[\%] = 30$.

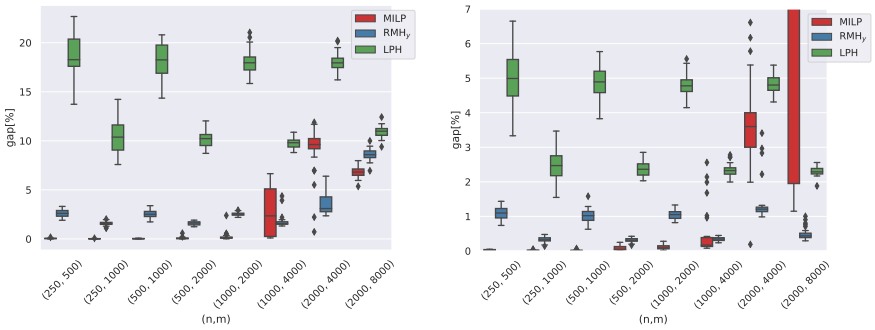
| | | MILP | | | $\text{RMH}_y$ | | | LPH | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | gap[%] | time[s] | $\lvert L(x)\rvert$ | gap[%] | time[s] | $\lvert L(x)\rvert$ | gap[%] | time[s] | $\lvert L(x)\rvert$ |
| 250 | 500 | 0.05 | 1800 | 25 | 2.61 | 91 | 25 | 18.62 | 2 | 81 |
| | 1000 | 0.02 | 1800 | 38 | 1.59 | 125 | 38 | 10.38 | 4 | 103 |
| 500 | 1000 | 0.03 | 1800 | 46 | 2.54 | 287 | 46 | 18.12 | 5 | 149 |
| | 2000 | 0.08 | 1800 | 72 | 1.60 | 686 | 71 | 10.06 | 12 | 190 |
| 1000 | 2000 | 0.24 | 1800 | 89 | 2.54 | 1295 | 88 | 17.95 | 20 | 279 |
| | 4000 | 2.69 | 1800 | 192 | 1.77 | 1800 | 129 | 9.78 | 47 | 346 |
| 2000 | 4000 | 9.09 | 1800 | 382 | 3.67 | 1800 | 166 | 18.01 | 81 | 532 |
| | 8000 | 6.78 | 1800 | 531 | 8.60 | 1800 | 535 | 10.92 | 238 | 660 |

(b) Results for $d_{\min}[\%] = 80$.

| | | MILP | | | $\text{RMH}_y$ | | | LPH | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | gap[%] | time[s] | $\lvert L(x)\rvert$ | gap[%] | time[s] | $\lvert L(x)\rvert$ | gap[%] | time[s] | $\lvert L(x)\rvert$ |
| 250 | 500 | 0.03 | 1800 | 47 | 1.09 | 47 | 47 | 4.98 | 2 | 86 |
| | 1000 | 0.02 | 1800 | 72 | 0.32 | 536 | 72 | 2.47 | 5 | 121 |
| 500 | 1000 | 0.02 | 1800 | 84 | 1.01 | 464 | 84 | 4.85 | 7 | 158 |
| | 2000 | 0.08 | 1800 | 138 | 0.31 | 1800 | 137 | 2.37 | 18 | 226 |
| 1000 | 2000 | 0.12 | 1800 | 160 | 1.04 | 1800 | 159 | 4.78 | 25 | 294 |
| | 4000 | 1.92 | 1800 | 305 | 0.35 | 1800 | 260 | 2.33 | 64 | 425 |
| 2000 | 4000 | 3.64 | 1800 | 488 | 1.40 | 1800 | 316 | 4.81 | 95 | 559 |
| | 8000 | 29.54 | 1800 | 1248 | 0.49 | 1800 | 515 | 2.31 | 236 | 815 |

less than 5 minutes on average. However, the derived feasible MBSSLP solutions are significantly worse than the solutions generated by $\text{RMH}_y$ especially for $d_{\min}[\%] = 30$. For instances nearly solved to optimally, we can also observe that the number of opened stations in the solutions are as expected. $\text{RMH}_y$ solutions require a marginally smaller number of opened stations than the MILP solutions. Solutions generated from the LPH approach, on the other hand, require a much higher number of opened stations than the other approaches. Hence, LPH does not seem to be a good choice as repair procedure for the LNS.

Figure 4 provides a more detailed comparison of the optimality gaps of the MILP, $\text{RMH}_y$ and LPH solutions. The figure shows boxplots of the optimality gaps for each instance group and approach and confirms our previous observations. Note that for a better comparison between the approaches Figure 4b is cut off and only shows optimality gaps up to 7% since solutions to the instances with $n = 1000, m = 4000$ as well as $n = 2000, m = 8000$ generated by the MILP feature optimality gaps up to 45%. For the largest instances with $n \geq 1000$ and $m \geq 4000$, $\text{RMH}_y$ starts to produce better results than the MILP while LPH does not seem to be able to compete with $\text{RMH}_y$ for any instance group. However, since $\text{RMH}_y$ requires solving a large MILP as well, this approach also has its limits concerning scalability. Therefore, in the next section we investigate the LNS that uses in each iteration $\text{RMH}_y$ to (re-)optimize only a comparably small part of a solution.




(a) Instances with $d_{\min}[\%] = 30$.

(b) Instances with $d_{\min}[\%] = 80$.

Fig. 4: Optimality gaps of the MILP, RMH$_y$ and LPH solutions.

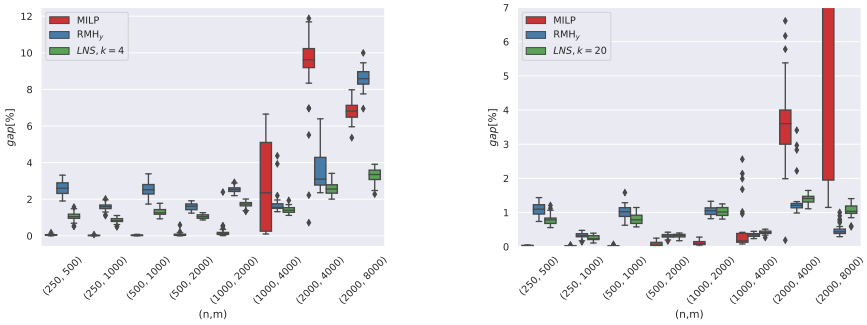
### 6.2 Large Neighborhood Search

For the size parameters of the repair operator we consider here, after preliminary tests $r = 4$ and $k \in \{4, 14, 20\}$. These values are promising as the MILPs corresponding to the repair subproblems can usually be solved to a small remaining optimality gap within seconds. As the LNS is a heuristic approach, it also does not make much sense to solve the MILPs always to proven optimality; instead we terminated the MILP solver when a solution with an optimality gap of at most 0.0005% has been reached. Each LNS run was terminated after 30 minutes. The results of the LNS are shown in Table 2. For each considered minimum demand coverage $d_{\min}$ and each neighborhood size parameter $k$, the average number of iterations "iter" and the average optimality gap "gap[%]" (w.r.t. the lower bounds obtained by the original MILP).

Table 2: Results of the LNS.

| | | $d_{\min}[\%] = 30$ | | | | | | $d_{\min}[\%] = 80$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | k=4 | | k=14 | | k=20 | | k=4 | | k=14 | | k=20 | |
| n | m | gap[%] | iter | gap[%] | iter | gap[%] | iter | gap[%] | iter | gap[%] | iter | gap[%] | iter |
| 250 | 500 | 1.05 | 2549 | 1.62 | 385 | 1.70 | 217 | 0.57 | 3222 | 0.75 | 520 | 0.79 | 270 |
| | 1000 | 0.83 | 1465 | 1.14 | 207 | 1.21 | 117 | 0.32 | 1843 | 0.23 | 263 | 0.25 | 117 |
| 500 | 1000 | 1.31 | 2094 | 1.64 | 418 | 1.83 | 207 | 0.72 | 2305 | 0.77 | 559 | 0.81 | 299 |
| | 2000 | 1.06 | 982 | 1.22 | 230 | 1.29 | 132 | 0.48 | 1115 | 0.33 | 282 | 0.31 | 156 |
| 1000 | 2000 | 1.72 | 1177 | 1.95 | 399 | 2.05 | 241 | 1.00 | 1375 | 1.02 | 482 | 1.03 | 317 |
| | 4000 | 1.41 | 604 | 1.44 | 203 | 1.45 | 132 | 0.78 | 606 | 0.46 | 214 | 0.42 | 128 |
| 2000 | 4000 | 2.58 | 698 | 2.64 | 292 | 2.69 | 211 | 1.59 | 720 | 1.46 | 331 | 1.39 | 251 |
| | 8000 | 3.28 | 306 | 3.10 | 128 | 3.06 | 93 | 2.00 | 280 | 1.11 | 128 | 1.06 | 87 |

The table shows that, naturally, the LNS can perform less iterations the larger $k$ is. For instances with $d_{\min}[\%] = 30$ we can see that the solutions tend to deteriorate as $k$ is increasing. However, this is not the case for instances with $d_{\min}[\%] = 80$ where we can see no such pattern. Moreover, as the instances become larger, the LNS with $k = 20$ starts to outperform the LNS with $k = 4$. Hence, for $d_{\min}[\%] = 80$ an LNS with even larger values for $k$ might yield better




(a) Instances with $d_{\min}[\%] = 30$.

(b) Instances with $d_{\min}[\%] = 80$.

Fig. 5: Comparison of the optimality gaps of the LNS solutions to the solutions of the other approaches.

results in theory. However, the larger $k$ is chosen the worse the scalability of the LNS becomes as the MILP that needs to be solved in the repair procedure takes longer to solve for larger values of $k$.

Figure 5 compares the optimality gaps of solutions obtained by the LNS to the optimality gaps of the MILP and RMH$_y$ solutions. Note that for a better comparison between the approaches Figure 5b is cut off and only shows optimality gaps up to 7%. For instances with $d_{\min}[\%] = 30$ we can see that the LNS is on average on all instance groups able to produce better solutions than RMH$_y$. This particularly holds for the largest instance group, where the gap of RMH$_y$ deteriorates to over 8% but the LNS' gaps are still within 4%. For instances with $d_{\min}[\%] = 80$, both, the LNS as well as RMH$_y$, perform quite well with gaps usually less than 2%. The LNS solutions are here slightly worse than the RMH$_y$ solutions for larger instances.

Overall, we can say that the LNS works reasonably well over all considered benchmark instances, and it is reasonable to expect it to scale much better to even larger instances than RMH$_y$ or solving the original MILP directly.

### 6.3 Results on the Manhattan Instance

In this section we show how well the MILP approaches as well as the LNS were able to deal with the real-world data based Manhattan instance. While the size of $n$ and $m$ is similar to some of our benchmark instances, the Manhattan instance is much harder to solve than our benchmark instances due to the shape of Manhattan as well as the instance's geographic distribution of demand.

Tables 3 and 4 show respective results. Each solution approach was applied to the instance six times with different values for $d_{\min}[\%]$. For each approach the tables lists the total costs of the solutions, the corresponding optimality gaps (always w.r.t. the lower bounds obtained from the linear relaxation of the original MILP), and the computation times in seconds. The direct MILP approach was only able to find (non optimal) solutions for the lowest levels of $d_{\min}[\%]$. RMH$_y$

and LPH could obtain feasible solutions for all cases except with $d_{\min}[\%] = 60$. Concerning $\text{RMH}_y$ and LPH, one can see that, as one might expect, gaps of LPH are usually significantly larger than those of $\text{RMH}_y$, but LPH is much faster and is, in contrast to $\text{RMH}_y$, also able to yield a feasible solution for $d_{\min}[\%] = 50$.

Table 4 shows the results obtained by the LNS with $r = 3$ and different values for $k$. Listed are total costs of the solutions, the corresponding optimality gaps (if a lower bound is known from the MILP), and the number of destroy and repair iterations. Most importantly, in contrast to the above MILP/LP approaches, the LNS could also find a feasible solution for $d_{\min}[\%] = 60$. Moreover, except for the lowest level of $d_{\min}[\%] = 10$, the LNS was able to find the best solutions. The number of performed destroy and repair iterations stays approximately the same for increasing levels of $d_{\min}[\%]$. However, as expected, the number of iterations decreases the larger the value for $k$.

Table 3: LPH, $\text{RMH}_y$, and MILP results for the Manhattan instance.

| | LPH | | | $\text{RMH}_y$ | | | MILP | | |
|---|---|---|---|---|---|---|---|---|---|
| $d_{\min}[\%]$ | costs | gap[%] | time[s] | costs | gap[%] | time[s] | costs | gap[%] | time[s] |
| 10 | 155797 | 1.27 | 179 | 153886 | 0.04 | 1801 | 153886 | 0.04 | 1801 |
| 20 | 325775 | 2.90 | 140 | 321773 | 1.69 | 1801 | 320168 | 1.20 | 1801 |
| 40 | 692976 | 1.06 | 196 | 689600 | 0.57 | 1801 | - | - | - |
| 50 | 892035 | 0.77 | 704 | - | - | - | - | - | - |
| 60 | - | - | - | - | - | - | - | - | - |

Table 4: LNS results for the Manhattan instance.

| | $k = 4$ | | | $k = 7$ | | | $k = 14$ | | |
|---|---|---|---|---|---|---|---|---|---|
| $d_{\min}[\%]$ | costs | gap[%] | iter | costs | gap[%] | iter | costs | gap[%] | iter |
| 10 | 153900 | 0.05 | 92 | 153890 | 0.05 | 19 | 154025 | 0.13 | 2 |
| 20 | 319769 | 1.07 | 87 | 319334 | 0.94 | 43 | 318939 | 0.82 | 19 |
| 40 | 688298 | 0.39 | 87 | 687769 | 0.31 | 42 | 687983 | 0.34 | 16 |
| 50 | 890049 | 0.55 | 83 | 888920 | 0.43 | 44 | 887926 | 0.32 | 24 |
| 60 | 1095190 | - | 89 | 1093898 | - | 43 | 1095097 | - | 15 |

## 7 Conclusions and Future Work

We presented the new Multi-Period Battery Swapping Station Location Problem (MBSSLP) for distributing battery swapping stations in an urban area. On our benchmark instances, directly solving the proposed MILP model is reasonable for instances with up to 1000 stations and 2000 O/D-pairs, where solutions with small gaps could be obtained. For larger instances solving the MILP model becomes quickly infeasible and heuristics need to be employed to find approximate solutions. Relaxing the $y$ variables and rounding obtained fractional values, i.e., our $\text{RMH}_y$, is a viable approach by which significantly larger instances can be solved reasonably well, nevertheless it also has its limits. We therefore proposed an LNS that effectively utilizes $\text{RMH}_y$ and provides better scalability. This can in particular be seen in the results for the real-world data based Manhattan instance.

We remark that the proposed LNS still has room for improvement. For example, different strategies for selecting the nodes to be removed or considered

for addition may be investigated. Moreover, adaptive mechanisms for choosing among different destroy and re-create methods may be useful. Last but not least, there are also alternative ways to address the scalability issue, for example by approaches based on (hierarchical) clustering and iterative refinement.

In future work the MBSSLP model should also be further refined to reflect real-world aspects in a more realistic way. For example, battery swapping stations are usually not extended slot by slot but by modules which consist of multiple new battery slots. So far, we also have not yet considered a pricing model for customers or costs for maintaining the battery swapping stations and the batteries.

## References


1. Cooper, L.: Location-allocation problems. Operations Research 11(3), 331–343 (1963)
2. Laporte, G., Nickel, S., da Gama, F.S. (eds.): Location science. Springer (2015)
3. Verter, V., Lapierre, S.D.: Location of preventive health care facilities. Annals of Operations Research 110(1), 123–132 (2002)
4. Berman, O., Larson, R.C., Fouska, N.: Optimal location of discretionary service facilities. Transportation Science 26(3), 201–211 (1992)
5. Kim, J.G., Kuby, M.: The deviation-flow refueling location model for optimizing a network of refueling stations. International Journal of Hydrogen Energy 37(6), 5406–5420 (2012)
6. Zarandi, M.H.F., Davari, S., Sisakht, S.A.H.: The large-scale dynamic maximal covering location problem. Mathematical and Computer Modelling 57(3), 710–719 (2013)
7. Hosseini, M., MirHassani, S., Hooshmand, F.: Deviation-flow refueling location problem with capacitated facilities: Model and algorithm. Transportation Research Part D: Transport and Environment 54, 269–281 (2017)
8. Kuby, M., Lim, S.: The flow-refueling location problem for alternative-fuel vehicles. Socio-Economic Planning Sciences 39(2), 125–145 (2005)
9. Upchurch, C., Kuby, M., Lim, S.: A model for location of capacitated alternative-fuel stations. Geographical Analysis 41(1), 85–106 (2009)
10. Mak, H.Y., Rong, Y., Shen, Z.J.M.: Infrastructure Planning for Electric Vehicles with Battery Swapping. Management Science 59(7), 1557–1575 (2013)
11. Zeng, M., Pan, Y., Zhang, D., Lu, Z., Li, Y.: Data-driven location selection for battery swapping stations. IEEE Access 7, 133760–133771 (2019)
12. Kuby, M.J., Kelley, S.B., Schoenemann, J.: Spatial refueling patterns of alternative-fuel and gasoline vehicle drivers in los angeles. Transportation Research Part D: Transport and Environment 25, 84–92 (2013)
13. Murali, P., Ordóñez, F., Dessouky, M.M.: Facility location under demand uncertainty: Response to a large-scale bio-terror attack. Socio-Economic Planning Sciences 46(1), 78–87 (2012), special Issue: Disaster Planning and Logistics: Part 1
14. Gendreau, M., Potvin, J.Y., et al.: Handbook of Metaheuristics, vol. 3. Springer (2019)
15. Capar, I., Kuby, M., Leon, V.J., Tsai, Y.J.: An arc cover–path-cover formulation and strategic analysis of alternative-fuel station locations. European Journal of Operational Research 227(1), 142–151 (2013)