A Multilevel Optimization Approach for Large Scale Battery Exchange Station Location Planning

Thomas Jatschka, Tobias Rodemann, Guenther Raidl

2023

Preprint:

This is a post-peer-review, pre-copyedit version of an article published in EvoCOP 2023. The final authenticated version is available online at: https://doi.org/10.1007/978-3-031-30035-6_4

The final publication is available at Springer via
https://doi.org/10.1007/978-3-031-30035-6_4.

A Multilevel Optimization Approach for Large Scale Battery Exchange Station Location Planning*

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

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

Abstract. We propose a multilevel optimization algorithm (MLO) for solving large scale instances of the Multi-Period Battery Swapping Station Location Problem (MBSSLP), i.e., a problem for deciding the placement of battery swapping stations in an urban area. MLO generates a solution to an MBSSLP instance in three steps. First the problem size is iteratively reduced by coarsening. Then, a solution to the coarsest problem instance is determined, and finally the obtained solution is projected to more fine grained problem instances in reverse order until a solution to the original problem instance is obtained. We test our approach on benchmark instances with up to 10000 areas for placing stations and 100000 user trips. We compare MLO to solving a mixed integer linear program (MILP) in a direct way as well as solving the instances with a construction heuristic (CH). Results show that MLO scales substantially better for such large instances than the MILP or the CH.

Keywords: multilevel optimization · mixed integer linear programming · E-mobility

1 Introduction

Electric vehicles (EVs) are becoming an increasingly popular way of transportation for the general public. However, a major inconvenience for the owners of an EV is the long time it takes to recharge a vehicle's battery. For smaller vehicles, such as electric scooters, a promising way to overcome this problem is to exchange a vehicle's battery instead of recharging it. Batteries of electric scooters are compact enough such that users can exchange their depleted batteries for fully charged ones at dedicated battery exchange stations within a short time and without assistance. At such stations batteries are recharged, and can later be provided to customers again.

We consider the Multi-Period Battery Swapping Station Location Problem (MBSSLP) as introduced in [3], where the setup costs for stations should be

* Thomas Jatschka acknowledges the financial support from the Honda Research Institute Europe.

minimized while a certain amount of customer demand needs to be satisfied. Each of the swapping stations is assumed to have a configurable number of slots at which batteries are charged and can be exchanged. Moreover, it is assumed that customers who want to change batteries specify their trip data (origin, destination, approximate time) online and are automatically assigned by the system to an appropriate station for the exchange (if one exists). As not every customer is willing to travel to a suggested station, e.g., if the detour is too long, the MBSSLP also considers a customer dropout which scales exponentially with the length of the detour induced by traveling to the assigned station.

In [3] a large neighborhood search (LNS) was presented for solving MBSSLP instances with up to roughly 2000 potential locations at which battery swapping stations can be placed and 8000 origin-destination (O/D) pairs that describe the customer trips. However, for real world applications especially the number of O/D-pairs can be magnitudes higher. The LNS proposed in [3] applies a destroy-and-repair scheme and uses a mixed integer linear programming based heuristic for repairing solutions. Unfortunately, this technique does not scale well to much larger instances. Finding good solutions for huge instances is in general a difficult task, even for metaheuristics, and one often resorts to clustering, refinement, or partitioning approaches that reduce the problem size or decompose the problem into smaller subproblems.

In this work we propose a multilevel optimization (MLO) approach for addressing large MBSSLP instances with tens of thousands of potential station areas and up to one hundred thousand user trips at which users need to swap batteries. The presented MLO is based on the algorithmic framework proposed by Walshaw [15], which has already been adapted to various mobility applications such as the traveling salesman problems [14], bike sharing station planning [7], and vehicle routing [10]. The basic idea of this approach is to generate a sequence of coarsened problem instances – referred to as multilevel coarsening – in which the problem sizes become successively smaller. After the coarsening process, a solution to the coarsest problem instance is generated. This solution is then iteratively projected to the problem instances of the multilevel coarsening in reverse order. Hence, after the final projection a solution to the original problem is obtained.

The underlying problem structure of the MBSSLP is given by a bipartite graph with one set representing the areas in which stations can be built and the other set representing the O/D-pairs. An MBSSLP instance is coarsened by first partitioning the nodes of the underlying graph and then deriving a coarsened graph by contracting the nodes in each partition. For solving the coarsest problem instances and for iteratively projecting solutions we make use of mixed integer linear programs (MILPs). Hereby, the MILP projecting a solution to a more detailed problem instance can be decomposed into subproblems, with each subproblem being responsible for projecting a single node of the underlying graph.

Our approach is experimentally evaluated on artificial benchmark scenarios generated as in [3]. To get a grasp of the quality of the solutions found by MLO

we compare the solutions to solutions generated by a MILP solver as well as solutions obtained from a construction heuristic for instances with up to tens of thousands potential station areas and up to one hundred thousand O/D-pairs. Results show that MLO scales substantially better than the MILP or the construction heuristic, generating reasonably good results in a short amount of time.

In the next Section we discuss related work. Section 3 provides a formal definition of the MBSSLP. Afterwards, in Section 4, we detail our MLO approach. Section 5 describes the benchmark instances and presents and discusses experimental results. Section 6 concludes this work with including outlook on promising future work.

2 Related Work

The MBSSLP, originally proposed in [3], is a capacitated multiple allocation facility location problem [8] and is loosely based on the capacitated deviation-flow refueling location model introduced in [2]. The MBSSLP also takes into account that not every customer is willing to travel to a predestined station when the detour is too long. Such customer satisfaction factors are modeled via a decay function as also done in, e.g., [13,6]. To the best of our knowledge, no multilevel optimization approach has yet been proposed for problems regarding the distribution of battery swapping stations or vehicle charging stations. In [5] a survey of charging station locations problems is provided. This review provides an overview of the size of instances which were considered in related work. From the approaches that also consider capacities of stations, a simulated annealing approach is described [16] to generate solutions for instances with up to 1400 potential station locations and about 15000 O/D-pairs. The instances for the MBSSLP in [3] contained up to 2000 potential locations and 8000 O/D-pairs.

The MLO framework for optimization problems was originally proposed by Walshaw [15] and has been applied to various applications, such as bike sharing [7], vehicle routing [10], or traveling salesman problems [14]. In [11] Valejo et al. give on overview of MLO approaches for complex networks.

3 The Multi-Period Battery Swapping Station Location Problem

The Multi-Period Battery Swapping Station Location Problem (MBSSLP) was originally proposed in [3]. We slightly modify the original MBSSLP formulation from [3] in certain aspects. First, we now consider a cyclic time horizon instead of a non-cyclic one as this appears to be more relevant in practice. More specifically, we assume a time horizon of one day that is discretized into equally long consecutive time intervals, for example hours. These intervals are indexed by $\mathcal{T} = \{0, \ldots, t_{\max} - 1\}$. As we consider the planning horizon to be cyclic the predecessor of the first interval is assumed to be the last one and the successor of the last one the first interval. Second, we generalize the problem in the sense

that instead of speaking of specific potential locations for service stations, we consider areas that may have more than one station. This extension is done in foresight of our MLO approach.

We assume battery swapping stations can be set up in any of n different areas referred to as set the L. Each area $l \in L$ has associated a maximum number of possible stations $r_l \in \mathbb{N}_{>0}$, a maximum number of possible battery charging slots $s_l > 0$ at each of these stations, fixed setup costs $c_l \geq 0$ for setting up one station in this area, and building costs per slot $b_l \geq 0$.

In contrast to some other work [1] that uses detailed multi-agent simulations to optimize system parameters, we model customers in an aggregated way as estimated travel demands in the form of a set of origin-destination (O/D) pairs (i.e., trips) Q and corresponding numbers $d_{qt} > 0$ of how often the need of swapping batteries is expected to arise within each time interval $t \in \mathcal{T}$ for each O/D-pair $q \in Q$. Let $m = |Q|$ be the number of O/D-pairs. As trips in an urban environment, as we consider it, are usually rather short, we assume for simplicity that trips start and end in the same time interval and can be completed with swapping batteries at most once.

Similarly to [6], we consider the satisfaction of users in dependence of detour lengths. Users will tend to avoid swapping batteries at trips for which detours to a swapping station are longer or for some other reason less convenient. To this end we associate each tuple (q, l) with $q \in Q, l \in L$ for each time interval $t \in \mathcal{T}$ with a value $g_{qlt} \in [0, 1]$ representing the satisfaction of customers. We make this factor also dependent on time as, e.g., in peak hours users are likely more hesitant to make a certain detour than in hours with not much traffic, respectively.

Let us now define the bipartite undirected graph $G = (Q, L, E)$, where the node sets Q and L correspond to the O/D-pairs and the areas for building swapping stations, respectively, Edge set $E \subseteq Q \times L$ shall include an edge (q, l) for each O/D-pair $q \in Q$ and area $l \in L$ whenever a swapping station with l could potentially satisfy (part of) the demand d_{qt}, i.e., $g_{qlt} > 0$ for at least one $t \in \mathcal{T}$. By $N(q) \subseteq L$, for $q \in Q$, we denote the set of adjacent nodes of node q, which corresponds to the subset of areas that are able to service O/D-pair Q. Vice versa, $N(l) \subset Q$, for any $l \in L$, denotes the adjacent nodes of the area node l, and thus, the O/D-pairs area l may service.

The number of time intervals required for completely recharging a battery is referred to as t^{c}. We make here the simplifying assumption that charging any battery always takes the same time and only completely recharged batteries are provided to customers again. We denote the set of time intervals in which a battery is not yet fully charged when returned to a station at time $t \in \mathcal{T}$ as $\mathcal{T}^{\mathrm{ch}}(t)$ which is defined as $\mathcal{T}^{\mathrm{ch}}(t) = \{((t + i) \bmod t_{\max}) \mid i = 0, \ldots, t^{\mathrm{c}}\}$.

In the original MBSSLP formulation a solution is feasible if a minimum amount of total customer demand $d_{\min}$ is satisfied. In foresight of our MLO approach we relax this condition in cases where $d_{\min}$ exceeds the total amount of demand that can be satisfied in any solution to G, referred to as $d_{\max}(G)$, and define a solution to be feasible if at least $\min(d_{\min}, d_{\max}(G))$ demand is satisfied.

For the development of MLO, we also store for each edge $(q,l) \in E(G)$ the maximum demand $\hat{d}_{qlt}$ that can be assigned from q to l in each time interval $t \in \mathcal{T}$, which is calculated by $\hat{d}_{qlt} = \min\left(\frac{\bar{d}_l}{g_{qlt}},\ d_{qt}\right)$ where $\bar{d}_l$ refers to the maximal necessary capacity of the stations in an area $l \in L$, i.e., $\bar{d}_l = \min\left(r_l s_l,\ \max_{t\in\mathcal{T}} \sum_{t'\in\mathcal{T}^{\mathrm{ch}}(t)} \sum_{q\in N(l)} g_{qlt'} d_{qt'}\right)$.

A solution to the MBSSLP is primarily given by a pair of vectors $x = (x_l)_{l\in L}$ with $x_l \in \{0,\dots,r_l\}$ and $y = (y_l)_{l\in L}$ with $y_l \in \{0,\dots,\lceil \bar{d}_l \rceil\}$, where x_l indicates the number of swapping stations to be established in area l and y_l represents the respective total number of battery slots at these stations. Moreover, a solution also has to specify which demand is fulfilled where. This is done by variables a_{qlt} that denote the part of d_{qt}, $q \in Q$, which is assigned to an area $l \in N(q)$ in time interval $t \in \mathcal{T}$. Customer satisfaction is considered by multiplying this assigned demand a_{qlt} with the factor g_{qlt} in order to obtain the actually fulfilled demand $\bar{a}_{qlt} = g_{qlt} a_{qlt}$ of O/D-pair q in area l in time interval t.

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

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

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

$$\bar{a}_{qlt} = g_{qlt} \cdot a_{qlt} \qquad t \in \mathcal{T},\ q \in Q,\ l \in N(q) \tag{3}$$

$$\sum_{l\in N(q)} a_{qlt} \leq d_{qt} \qquad t \in \mathcal{T},\ q \in Q \tag{4}$$

$$\sum_{t'\in\mathcal{T}^{\mathrm{ch}}(t)} \sum_{q\in N(l)} \bar{a}_{qlt'} \leq y_l \qquad t \in \mathcal{T},\ l \in L \tag{5}$$

$$\sum_{t\in\mathcal{T}} \sum_{q\in Q} \sum_{l\in N(q)} \bar{a}_{qlt} \geq \min(d_{\min}, d_{\max}(G)) \tag{6}$$

$$x_l \in \{0,\dots,r_l\} \qquad l \in L \tag{7}$$

$$y_l \in \{0,\dots,\lceil \bar{d}_l \rceil\} \qquad l \in L \tag{8}$$

$$0 \leq a_{qlt} \leq \hat{d}_{qlt} \qquad t \in \mathcal{T},\ q \in Q,\ l \in N(q) \tag{9}$$

$$0 \leq \bar{a}_{qlt} \leq g_{qlt} \hat{d}_{qlt} \qquad t \in \mathcal{T},\ q \in Q,\ l \in N(q) \tag{10}$$

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 an area $l \in L$ if a sufficient number of stations is opened there. Equalities (3) calculate fulfilled demands $\bar{a}_{qlt}$ by applying the customer satisfaction factors g_{qlt} to the assigned demands a_{qlt}. Constraints (4) enforce that the total demand assigned from an O/D-pair q to areas does not exceed d_{qt} for all $t \in \mathcal{T}$. Inequalities (5) ensure that the capacity y_l is not exceeded at all areas over all time intervals. Note that by using $\bar{a}_{qlt}$ instead of a_{qlt} in (5), we "overbook" areas to consider the expected case, similarly as in [9]. Inequalities (5) also model that swapped batteries cannot be

reused for the next t^c time intervals in which they are being charged again. 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)–(10). Note that $d_{\max}(G)$ can be calculated by replacing the objective function (1) with

$$\max \sum_{t \in \mathcal{T}} \sum_{q \in Q^K} \sum_{l \in N(q)} g_{qlt} \cdot a_{qlt} \tag{11}$$

and removing Constraint (6).

4 Multilevel Refinement Algorithm

Our multilevel optimization approach (MLO) follows the basic scheme proposed by [15] and consists of three steps: iteratively coarsening the problem instance by partitioning and contraction, solving the coarsest instance, and iteratively uncoarsening by projection and possible refinement. During the coarsening step the problem complexity is iteratively reduced by merging areas for setting up stations and O/D-pairs until the size of the problem instance falls below a certain threshold. Then, a solution to the coarsest instance is generated. Afterwards, this solution is successively extended by projecting it to the less coarsened instances and refining it, eventually resulting in a feasible solution to the original instance.

We define a multilevel coarsening for the MBSSLP by the graph sequence $\{G^0, \ldots, G^K\}$ of G with $G^i = (Q^i, L^i, E^i)$, for $i = 0, \ldots, K$. The graph on the lowest level corresponds to the original graph G, i.e., $G^0 = G$. As the original problem graph G, each graph G^i also have respective associated values r_l, s_l, c_l, b_l, and $\bar{d}_l$ for the nodes $l \in L^i$, values d_{qt} for nodes $q \in Q^i$, and values g_{qlt} and $\hat{d}_{qlt}$ for the edges $(q, l) \in E^i$ and $t \in \mathcal{T}$. A graph G^{i+1} with $i \in \{0, \ldots, K-1\}$ is derived from G^i by partitioning Q^i and L^i and merging all nodes within each partition. The vertices $q \in Q^{i+1}$ and $l \in L^{i+1}$ are associated with a non-empty subset of Q^i and L^i, denoted as Q^i_q and L^i_l, referring to the respective partitions of G^i. Hence, it must hold that $Q^i_q \cap Q^i_{q'} = \emptyset$ and $L^i_l \cap L^i_{l'} = \emptyset$ for any $q, q' \in Q^i,\ q \neq q'$ and $l, l' \in L^i,\ l \neq l'$, and $i = 1, \ldots, K$. Finally, (q, l) is an edge in E^{i+1} if there is at least one edge between the nodes Q^i_q and L^i_l in G^i.

Algorithm 1 shows our MLO approach in pseudo-code. Note that in our approach it cannot be guaranteed that a solution obtained for a graph G^{i+1} can be projected to G^i such that the projected solution satisfies at least the same amount of demand as the previous solution. Therefore, after projecting a solution to a graph G^i, we refine it to increase the amount of satisfied demand until the solution becomes feasible w.r.t. G^i.

In the following we describe the concrete steps of our MLO approach in more detail.

Partitioning. A graph G^{i+1} is derived from G^i by first partitioning the node sets of G^i and then contracting the nodes within each partition. For deriving the partitioning of our bipartite graphs we use the same approach as proposed by Valejo et al. [12]. We first generate two unipartite graphs for each vertex set

Algorithm 1: MLO

Input : an MBSSLP instance, the number of coarsening steps K
Output: a solution (x, y, a)

$i \leftarrow 0$;
$G^i \leftarrow G$;
while $i < K$ **do**
    $G^{i+1} \leftarrow \text{coarsen}(G^i)$;
    $i \leftarrow i + 1$;
end while
$(x, y, a) \leftarrow$ solve problem w.r.t. G^i; // coarsest problem instance
while $i > 0$ **do**
    $(x, y, a) \leftarrow$ project solution (x, y, a) for G^i to a solution for G^{i-1};
    $(x, y, a) \leftarrow$ refine solution (x, y, a);
    $i \leftarrow i - 1$;
end while
return (x, y, a);

L^i, Q^i of G^i via one-mode projection, i.e., the vertices of these unipartite graphs are given by the vertices of the corresponding vertex set of G^i with two vertices being adjacent if they have common neighbors in G^i. For calculating weights between two nodes of u, v of L^i or Q^i, respectively, we use the Jaccard similarity measure

$$\chi(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}. \tag{12}$$

Afterwards, each unipartite graph is partitioned independently via *greedy heavy-edge-matching* (GHEM) [11]. GHEM is a variation of *heavy-edge matching* [4]. The GHEM heuristic iterates over all edges of a graph in descending order and in every iteration partitions the incident nodes of the current edge and removes them from the graph.

Contracting. Recall that we denote a partition of nodes of G^{i-1} as Q^{i-1}_q and L^{i-1}_l, respectively, with $q \in Q^i, l \in L^i$, for $i = 1, \ldots, K$.

When contracting nodes, one has to also aggregate the associated node and edge properties in meaningful ways. To facilitate the later projection of solutions, we split the coarsening of G^{i-1} into two steps, first deriving from G^{i-1} an intermediate graph $\tilde{G}^i$, in which only the partitions on L^{i-1} are merged, and then from $\tilde{G}^i$ the actual G^i in which also the partitions on Q^{i-1} are merged. In the following we denote the node and edge sets of $\tilde{G}^i$ as well as all associated values correspondingly with a tilde, i.e., $\tilde{G}^i = (\tilde{Q}^i, \tilde{L}^i, \tilde{E}^i)$. Moreover, $\tilde{N}^i(\cdot)$ refers to adjacent nodes of nodes in $\tilde{G}^i$, whereas $N^i(\cdot)$ refers only to neighbors of nodes in G^i.

To obtain $\tilde{G}^i$, we thus directly adopt $\tilde{Q}^i = Q^{i-1}$ together with the properties associated with these nodes, i.e., the respective demands. The partitions on L^{i-1} on the other hand are merged to obtain the new area node set $\tilde{L}^i$.

Maximum allowed demand assignments for $(q,l) \in \tilde{E}^i$ pairs are now calculated as

$$\hat{d}_{qlt} = \min\left(\sum_{l' \in L_l^{i-1}} \hat{d}_{ql't},\ d_{qt}\right), \tag{13}$$

and the customer satisfaction factors are determined as weighted average

$$g_{qlt} = \frac{\sum_{l' \in L_l^{i-1}} \hat{d}_{ql't} \cdot g_{ql't}}{\sum_{l' \in L_l^{i-1}} \hat{d}_{ql't}}. \tag{14}$$

The maximum demand that can be fulfilled by the stations in an area $l \in \tilde{L}^i$ in any time interval is determined by

$$\bar{d}_l = \min\left(\sum_{l' \in L_l^{i-1}} \bar{d}_{l'},\ \max_{t \in \mathcal{T}} \sum_{t' \in \mathcal{T}^{\mathrm{ch}}(t)} \sum_{q \in \tilde{N}^i(l)} g_{qlt'} \hat{d}_{qlt'}\right). \tag{15}$$

For each $l \in \tilde{L}^i$, s_l, c_l, and b_l are averaged in a weighted manner:

$$s_l = \left\lceil \frac{\sum_{l' \in L_{l'}^{i-1}} \bar{d}_{l'} s_{l'}}{\sum_{l' \in L_l^{i-1}} \bar{d}_{l'}} \right\rceil, \quad c_l = \frac{\sum_{l' \in L_l^{i-1}} \bar{d}_{l'} c_{l'}}{\sum_{l' \in L_l^{i-1}} \bar{d}_{l'}}, \quad b_l = \frac{\sum_{l' \in L_l^{i-1}} \bar{d}_{l'} b_{l'}}{\sum_{l' \in L_l^{i-1}} \bar{d}_{l'}}. \tag{16}$$

Finally, maximum station numbers are derived by

$$r_l = \left\lceil \frac{\bar{d}_l}{s_l} \right\rceil. \tag{17}$$

To finally obtain G^i from the intermediate $\tilde{G}^i$, we directly adopt $L^i = \tilde{L}^i$ together with all the properties associated with these nodes, while merging the partitions on $\tilde{Q}^i$ obtaining the new O/D-pair node set Q^i.

Customer satisfaction factors are aggregated again by taking the weighted average

$$g_{qlt} = \frac{\sum_{q' \in \tilde{Q}_q^{i-1}} \hat{d}_{q'lt} \cdot g_{q'lt}}{\sum_{q' \in \tilde{Q}_q^i} \hat{d}_{q'lt}}. \tag{18}$$

For each edge $(q,l) \in \tilde{E}^i$ and each $t \in \mathcal{T}$, the maximum assignable demand is calculated by

$$\hat{d}_{qlt} = \min\left(\sum_{q' \in \tilde{Q}_q^i} \hat{d}_{q'lt},\ \frac{\bar{d}_l}{g_{qlt}}\right). \tag{19}$$

The demands of these O/D-pairs are aggregated by taking the respective sums for all $q \in \tilde{Q}^i$ and $t \in \mathcal{T}$ while also considering the maximal amount of demand d_{qlt} that can be assigned to stations at all adjacent areas $l \in N^i(q)$:

$$d_{qt} = \min\left(\sum_{q' \in \tilde{Q}_q^i} d_{q't},\ \sum_{l \in N^i(q)} \hat{d}_{qlt}\right). \tag{20}$$

Note that due to the aggregation of the customer satisfaction factors g, it cannot be guaranteed that $d_{\max}(G^i) \geq d_{\min}$. Therefore, as previously discussed we have relaxed the original feasibility criterion of the MBSSLP such that a solution to a graph G^i is feasible if at least $\min(d_{\min}, d_{\max}(G^i))$ demand is satisfied.

Solving the Coarsest Graph. The MILP (1)–(10) is used to generate a solution to the coarsest graph G^K.

Projecting. A solution to a graph G^i is projected to the graph G^{i-1} in two steps. First the solution is projected to $\tilde{G}^i$ and then further projected to G^{i-1}.

Let x, y, and a be defined as described in Section 3. Projecting a solution from G^i to $\tilde{G}^i$ is done by solving a linear program (LP) for each $q \in Q^i$:

$$\max \sum_{q' \in Q_q^i} \sum_{l \in \tilde{N}^i(q)} \sum_{t \in \mathcal{T}} \frac{g_{q'lt} a_{q'lt}}{b_l} \tag{21}$$

$$\sum_{l \in \tilde{N}^i(q')} a_{q'lt} \leq d_{q't} \qquad t \in \mathcal{T},\ q' \in Q_q^{i-1} \tag{22}$$

$$\sum_{q' \in \tilde{N}^i(l) \cap Q_q^i} g_{q'lt} \cdot a_{q'lt} \leq g_{qlt} \cdot a_{qlt} \qquad t \in \mathcal{T},\ l \in N^i(q) \tag{23}$$

$$0 \leq a_{q'lt} \leq \hat{d}_{q'lt} \qquad t \in \mathcal{T},\ q' \in Q_q^i,\ l \in \tilde{N}^i(q') \tag{24}$$

Recall that variables a_{qlt} denote the part of d_{qt} w.r.t. $q \in Q$, which is assigned to an area $l \in N(q)$ in time interval $t \in \mathcal{T}$. The objective function of this LP maximizes the ratio of assigned demand to costs for building modules at the respective areas. Constraints (22) ensure that assigned demand does not exceed an O/D-pair's available demand. Constraints (23) ensure that the total demand assigned from all $q' \in Q_q^{i-1}$ to some $l \in \tilde{L}^i$ does not exceed the demand assigned from q to l. Hence, the total number of battery slots required in an area does not increase when projecting the solution to $\tilde{G}^i$. Note that the sub-problems induced by $q \in Q^i$ can be solved independently of each other. However, the total satisfied demand for the obtained solution might be smaller than the satisfied demand in the solution to G^i.

When the solution is projected from $\tilde{G}^i$ to G^{i-1}, we again have one sub-problem for each $l \in \tilde{L}^i$. In this step we also aim to compensate for satisfied demand lost in previous solution projections. Let d_{missing} denote the difference between $d_{\min}$ and the amount of demand satisfied in a solution and let $\delta_{\min}(l) = \sum_{q \in \tilde{N}^i(l)} g_{qlt} \cdot a_{qlt} + d_{\text{missing}}$ for $l \in \tilde{L}^i$. Then, when projecting the solution w.r.t. l, the minimal amount of demand to be satisfied by the areas in L_l^i is $\min(\delta_{\min}(l), d_{\max}(L_l^{i-1}))$ where $d_{\max}(L_l^{i-1})$ is the maximal amount of demand that can be satisfied by the areas in L_l^{i-1}. Moreover, when projecting the solution w.r.t. $\tilde{L}^i$ we not only consider the demand allocated at the areas in the solution but also take into account the so far unassigned demand of all O/D-pairs in $\tilde{Q}^i$. Hence, a sub-problem for $l \in \tilde{L}^i$ induces a sub-instance with the areas L_l^{i-1}, the

Algorithm 2: Project Solution from $\tilde{G}^i$ to G^{i-1}

Input : a solution (x, y, a) to $\tilde{G}^i$
Output: a solution to G^{i-1}

$\Lambda \leftarrow \{l \in \tilde{L}^i \mid x_l > 0\}$; //areas to be extended
$\rho \leftarrow \left(\frac{\sum_{q \in \tilde{N}^i(l)} g_{qlt} \cdot a_{qlt}}{x_l \cdot c_l + y_l \cdot b_l} \right)_{l \in \Lambda}$;
$\delta_{\min} \leftarrow d_{\min} - \sum_{t \in \mathcal{T}, l \in \Lambda, q \in \tilde{N}^i(l)} a_{qlt}$;
$\delta \leftarrow \left(d_{qt} - \sum_{l \in \tilde{N}^i(q)} a_{qlt} \right)_{q \in \tilde{Q}^i, t \in \mathcal{T}}$; //unassigned demand
while $|\Lambda| > 0$ **do**
  if $\delta_{\min} > 0$ **then**
    $l \leftarrow \arg\max_{l \in \Lambda} \{\rho_l\}$;
  else
    $l \leftarrow \arg\min_{l \in \Lambda} \{\rho_l\}$;
  end if
  $\Lambda \leftarrow \Lambda \setminus \{l\}$;
  $\delta_{\min} \leftarrow \delta_{\min} + \sum_{q \in \tilde{N}^i(l)} g_{qlt} \cdot a_{qlt}$;
  $\delta_{qt} \leftarrow \delta_{qt} + a_{qlt} \quad \forall q \in \tilde{N}^i(l),\ t \in \mathcal{T}$;
  $(x', y', a') \leftarrow \text{solve}(L_l^{i-1}, \tilde{N}^i(l), \delta_{\min}, \delta)$; //apply MILP (1)–(10)
  for $l' \in L_l^{i-1}$ **do**
    $x_{l'} \leftarrow x'_{l'},\ y_{l'} \leftarrow y'_{l'}$;
    for $q \in N^{i-1}(l'),\ t \in \mathcal{T}$ **do**
      $a_{ql't} \leftarrow a'_{ql't}$;
      $\delta_{\min} \leftarrow \delta_{\min} - g_{ql't} \cdot a_{ql't}$;
      $\delta_{qt} \leftarrow \delta_{qt} - a_{ql't}$;
    end for
  end for
end while
return (x, y, a);

O/D-pairs $q \in \tilde{N}^{i-1}(l)$ with available demands δ_{qt} for $t \in \mathcal{T}$, and a minimal amount of demand to be satisfied $\delta_{\min}(l)$. This sub-instance can then be solved by the MILP (1)–(10). Algorithm 2 gives a detailed description of how δ and $\delta_{\min}(l)$ are calculated for each sub-problem. Note that these sub-problems are no longer independent of each other. Therefore, the order in which they are solved impacts the quality of the projected solution. To keep the setup costs of the projected solution as low as possible, Algorithm 2 chooses the sub-problem induced by the most cost efficient area $l \in \tilde{L}^i$ if d_{missing} is greater than zero as the next one. Otherwise, if the current solution satisfies $d_{\min}$ demand, the sub-problem induced by the least cost efficient area is solved next.

Algorithm 3: Refine Solution

Input : a solution (x, y, a) to G^i
Output: a refined solution to G^i

$\Lambda \leftarrow L^i$;
$\rho \leftarrow \left((r_l - x_l) \cdot c_l + (\lceil \bar{d}_l \rceil - y_l) \cdot b_l\right)_{l \in \Lambda}$;
$\delta_{\min} \leftarrow d_{\min} - \sum_{t \in \mathcal{T}, l \in \Lambda, q \in N^i(l)} a_{qlt}$;
$\delta \leftarrow \left(d_{qt} - \sum_{l \in N(q)} a_{qlt}\right)_{q \in \tilde{Q}^i, t \in \mathcal{T}}$; //unassigned demand
while $\delta_{\min} > 0$ **do**
  $l \leftarrow \arg\min_{l \in \Lambda}\{\rho_l\}$;
  $\Lambda \leftarrow \Lambda \setminus l$;
  $\delta_{\min} \leftarrow \delta_{\min} + \sum_{q \in N^i(l)} g_{qlt} \cdot a_{qlt}$;
  $\delta_{qt} \leftarrow \delta_{qt} + a_{qlt} \quad \forall q \in N^i(l),\ t \in \mathcal{T}$;
  $(x', y', a') \leftarrow \text{solve}(l, N^i(l), \delta_{\min}, \delta)$; //apply MILP (1)–(10)
  $x_l \leftarrow x'_l$, $y_l \leftarrow y'_l$;
  for $q \in N(l'),\ t \in \mathcal{T}$ **do**
    $a_{qlt} \leftarrow a'_{qlt}$;
    $\delta_{\min} \leftarrow \delta_{\min} - g_{qlt} \cdot a_{qlt}$;
    $\delta_{qt} \leftarrow \delta_{qt} - a_{qlt}$;
  end for
end while
return (x, y, a);

Refine Solution. After projecting a solution from $\tilde{G}^{i+1}$ to G^i via Algorithm 2, the obtained solution may be infeasible as it cannot be guaranteed that a solution to a sub-problem w.r.t. $l \in \tilde{L}^i$ can actually satisfy $\delta_{\min}(l)$ demand. Therefore, it may be necessary to further refine the obtained solution, i.e., to open additional modules or areas and assign demand to them. Our refinement procedure is shown in Algorithm 3. Similar to Algorithm 2 for each $l \in L^i$ that is not yet fully utilized we define a sub-instance with area l, O/D-pairs $q \in N^i(l)$ with available demands δ_{qt} for $t \in \mathcal{T}$, and a minimal amount of demand to be satisfied $\delta_{\min}(l)$. Again, this sub-instance can be solved by the MILP (1)–(10). As long as the solution is infeasible, the sub-instance induced by the cheapest area is chosen and solved. Note that we use Algorithm 3 also as standalone construction heuristic (CH) and will compare the performance of MLO to the performance of CH.

5 Computational Results

We test our approach on artificial instances generated as described in [3]. A total of eight groups of instances identified by their number of station areas n and number of O/D pairs m as (n, m) is created. Each group contains 30 instances. Note that the instances contain some station areas that have no O/D pairs in the vicinity and vice versa. These unconnected nodes are deleted during preprocessing. Table 1 gives an overview over all instance groups. Columns n_{pp} and

Table 1: Test instance groups and the average numbers of nodes after preprocessing.

n	m	n_{pp}	m_{pp}
5000	5000	2354	4936
	12500	3086	12410
	25000	3663	24821
	50000	4194	49634

n	m	n_{pp}	m_{pp}
10000	10000	4658	9457
	25000	6130	24845
	50000	7300	49714
	100000	8394	99407

Table 2: Results obtained by solving the MILP with Gurobi (runtime limit two hours).

n	m	γ_{LB}[%]	n_{feasible}	τ[s]
5000	5000	3.47	30	7200
	12500	12.01	30	7200
	25000	6.40	25	7200
	50000	4.41	7	7202
10000	10000	21.79	30	7200
	25000	13.77	24	7201
	50000	-	0	-
	100000	-	0	-

m_{pp} list the average numbers of station areas and O/D pairs, respectively after preprocessing. Note that the number of actually usable station areas strongly depends on the number of O/D pairs, i.e., the more O/D pairs a graph with a fixed number of station areas has, the more station areas are adjacent to an O/D pair.

MLO was implemented in Julia[3] 1.8.1 using Gurobi[4] 9.1 as the underlying MILP solver. All test runs have been executed on an AMD EPYC 7402, 2.80GHz machine in single-threaded mode with a global memory limit of 100GB. When solving MILPs during MLO we have set a time limit of ten minutes and terminated the solving earlier when an optimality gap of $\leq 0.5\%$ was reached.

First, Table 2 shows the results for generating solutions to the benchmark instances by solving the MILP (1)–(10) with Gurobi with a runtime limit of two hours. Column γ_{LB} shows average gaps between the best found solutions and the respective lower bounds. Additionally, column n_{feasible} shows for how many instances in a group the solver was able to find a feasible solution. Finally, column τ shows median computation times. We can see that the MILP solver was not able to consistently find a feasible solution within the given time limit for a large part of the instances. In general, the larger the instances, the fewer solutions were found by the MILP solver. However, there are three groups for which the MILP solver was able to find a feasible solution to all instances:

[3] `https://julialang.org/`
[4] `https://www.gurobi.com/`

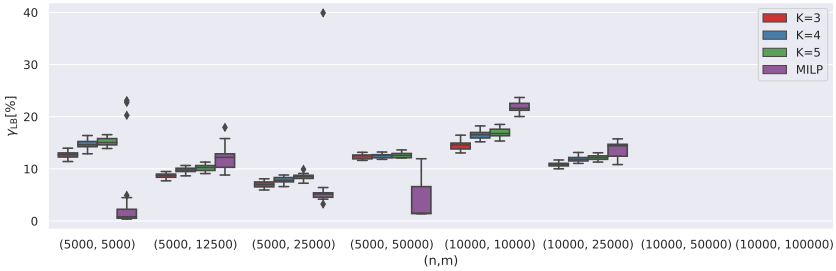

Fig. 1: Comparison of results obtained by a MILP solver to results obtained by MLO with different values for K. Each boxplot covers a single instance group and each instance was evaluated once for each configuration.

$(5000, 5000)$, $(5000, 12500)$, and $(10000, 10000)$. Looking at these groups we can also see that the gaps to the best known lower bounds are strongly increasing as the instances become larger. Moreover, for the two largest instance groups the MILP solver was not able to find any feasible solution at all within the time limit.

Next, in Figure 1 we compare different variants of MLO to the pure MILP approach. MLO was able to find feasible solutions for all instances, however, for this comparison we only consider instances to which MLO as well as the MILP solver were able to find solutions. To compare the two approaches we use the best found lowest bound for each instance reported by Gurobi to calculate gaps for the MLO results. We test MLO with different numbers of coarsening levels $K \in \{3, 4, 5\}$. The figure shows that for the reported instances, MLO achieves gaps between 6% to 18%. For the instance groups $(5000, 12500)$, and $(10000, 10000)$, for which the MILP solver found solutions to all instances and the group $(10000, 25000)$, MLO is able to achieve better results than the MILP. Otherwise, the MILP solutions are usually better than the MLO solutions. However, one has to consider, that the MILP solver was not able find solutions to a large part of the instances at all. Additionally, the MILP solver terminated after two hours, while MLO was much faster.

To get a better impression of how well MLO performs, Tables 3–5 give more detailed results for MLO. As previously mentioned the construction heuristic used for refining solutions after projection can also be used as a standalone construction heuristic (CH). We compare the results obtained by MLO to the results achieved with this construction heuristic as MLO itself is comparable to a construction heuristic in the sense that a complete solution is only obtained at the end of the algorithm. Moreover, as CH generates solutions to all instances within the time limit, we are able to get a better impression of how well MLO performs w.r.t. different values of K and different instance sizes. Columns γ_{CH} show the average gap between the MLO results and the CH results for each instance group. More specifically, let f_{MLO} refer to the objective value of a solution obtained with MLO and let f_{CH} refer to the objective value of a solution

Table 3: MLO results for $K = 3$.

n	m	γ_{CH}[%]	n_{proj}	τ_{c}[s]	τ_{cp}[s]	τ[s]
5000	5000	-37.38	5279	6	36	103
	12500	-48.54	13032	12	123	265
	25000	-36.84	24688	28	96	385
	50000	-22.09	47304	77	81	663
10000	10000	-37.62	10156	10	132	241
	25000	-48.93	26035	24	457	747
	50000	-36.88	49348	58	358	954
	100000	-22.09	94635	173	292	1624

Table 4: MLO results for $K = 4$.

n	m	γ_{CH}[%]	n_{proj}	τ_{c}[s]	τ_{cp}[s]	τ[s]
5000	5000	-34.30	5950	6	20	86
	12500	-46.85	14275	11	14	154
	25000	-35.68	26764	26	12	277
	50000	-21.63	51073	71	17	552
10000	10000	-34.32	11466	9	39	149
	25000	-47.21	28480	22	50	325
	50000	-35.59	53482	54	41	603
	100000	-21.46	102171	160	60	1301

Table 5: MLO results for $K = 5$.

n	m	γ_{CH}[%]	n_{proj}	τ_{c}[s]	τ_{cp}[s]	τ[s]
5000	5000	-33.50	6276	6	3	68
	12500	-46.12	14920	11	2	139
	25000	-34.71	27870	25	3	271
	50000	-21.44	53064	70	5	543
10000	10000	-33.67	12113	9	8	117
	25000	-46.75	29759	21	6	281
	50000	-34.76	55703	51	9	576
	100000	-21.42	106172	155	15	1255

(to the same instance) generated by CH. Then, $\gamma_{\text{CH}} = \frac{f_{\text{CH}} - f_{\text{MLO}}}{f_{\text{CH}}} \cdot 100$. Hence, if γ_{CH} is negative, MLO achieved better results, otherwise the better result was achieved by CH. Additionally, the tables also show the average number of sub-problems n_{proj} solved when projecting the solution, the average time τ_{c} needed for generating all coarsened graphs, the average time τ_{cp} for generating a solution to the coarsest graph, as well as the average total computation time τ. Results indicated that MLO clearly outperforms CH w.r.t. the obtained solution quality, yielding solutions that are up to $\approx 49\%$ better on average. Note that the gaps generally decrease as m increases. Due to a higher number of O/D pairs, a larger number of opened stations is required. Hence, the difference between the worst solution (the solution in which all stations are opened with a maximum number of modules) and an optimal solution becomes smaller. Therefore, one can also expect to achieve "better" results more easily as m increases. As expected, one can also observe that the number of sub-problems that need to be solved during the projection phase increases proportionally to K. However, in general the number of sub-problems solved is roughly equal to m. As the number of sub-problems increases, the total computation times actually decrease as the coarsest problem instance can be solved significantly faster for larger values of K. In general, it seems that the computation times grow proportional to n and m. For the largest instance group MLO needed up to 27 minutes on average to generate solutions. Moreover, coarsening of the graphs contributes only a small part of the total computation time. Finally, we can also observe that the solution quality slightly deteriorates as K increases. This behavior is not surprising as the

coarsest instance becomes a less accurate representation of the original instance the more often it was coarsened.

6 Conclusion and Future Work

We presented a Multilevel Optimization (MLO) approach for solving huge instances of the Multi-Period Battery Swapping Station Location Problem (MBSSLP) proposed in [3]. MLO first generates series of coarsened graphs with each new graph becoming smaller in size. Afterwards, a solution to the coarsest graph is generated and iteratively projected to the coarsened graphs in reverse order until and refined a solution to the original problem graph is obtained.

The approach was tested on artificial benchmark instances with up to 10000 areas for placing stations and 100000 origin-destination (O/D) pairs describing the trips of users. Evaluating our approach on these benchmark instances shows that MLO is able to generate reasonably good solutions within at most half an hour. On the other hand, when formulating and solving the problem as a mixed integer linear program (MILP), the MILP solver struggles to consistently find even feasible solutions to many of the instances. As the size of the instances increases, MLO clearly outperform the MILP solver w.r.t. the achieved solution quality. When confronted with such huge instances, classical metaheuristics often struggle as well to obtain good results as corresponding neighborhood structures are too large to be searched efficiently. Therefore, we instead compared MLO to the construction heuristic (CH) that was used within MLO for refining solutions. Our results show that MLO significantly outperforms CH w.r.t. all instances.

Still, there are multiple ways in which MLO can possibly be improved, especially w.r.t. deriving the coarsened graphs. Using greedy heavy-edge matching we obtain a graph partitioning that contains at most two nodes in each partition. However, it seems promising to adapt this approach such that partitions of larger size can be generated if, for example, one can identify large similarities between multiple nodes. In order to identify similar nodes more reliably it seems also necessary to derive a more problem specific similarity criterion. A possible way to achieve this is to use machine learning for learning the similarity between two nodes, e.g., by training a machine learning model on a smaller set of instances in which only two nodes are coarsened at a time.

A particular issue for MLO is that the amount of satisfied demand can decrease when projecting a solution. A potential way to alleviate this problem is to derive constraints that restrict the possibilities in how a node can be projected.

In this contribution we have explicitly specified how often a graph should be coarsened. Additionally, in each iteration both the set of station areas as well as the set of O/D-pairs is coarsened. In future work we aim to adapt MLO such that a graph is coarsened until both of its vertex sets have in some sense ideal sizes.

Finally, in the real world a solution obtained by MLO is in general not build at once but gradually over time. Therefore, an interesting further problem is to also optimize the schedule by which the stations should be built when realizing the obtained solution.

References

1. Brulin, S., Bujny, M., Puphal, T., Menzel, S.: Data-driven evolutionary optimization of eVTOL design concepts based on multi-agent simulations. In: Proceedings of the American Institute of Aeronautics and Astronautics SciTech Forum (to appear)
2. 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)
3. Jatschka, T., Oberweger, F.F., Rodemann, T., Raidl, G.R.: Distributing battery swapping stations for electric scooters in an urban area. In: Olenev, N., Evtushenko, Y., Khachay, M., Malkova, V. (eds.) Optimization and Applications, Proceedings of OPTIMA 2020 – XI International Conference Optimization and Applications. LNCS, vol. 12422, pp. 150–165. Springer (2020)
4. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing **20**(1), 359–392 (1998)
5. Kchaou-Boujelben, M.: Charging station location problem: A comprehensive review on models and solution approaches. Transportation Research Part C: Emerging Technologies **132** (2021)
6. 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)
7. Kloimüllner, C., Raidl, G.R.: Hierarchical clustering and multilevel refinement for the bike-sharing station planning problem. In: Battiti, R., Kvasov, D.E., Sergeyev, Y.D. (eds.) Learning and Intelligent Optimization. LNCS, vol. 10556, pp. 150–165. Springer International Publishing, Cham (2017)
8. Laporte, G., Nickel, S., da Gama, F.S. (eds.): Location science. Springer (2015)
9. 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
10. Pirkwieser, S., Raidl, G.R.: Multilevel variable neighborhood search for periodic routing problems. In: Cowling, P., Merz, P. (eds.) Evolutionary Computation in Combinatorial Optimization. LNCS, vol. 6022, pp. 226–238. Springer (2010)
11. Valejo, A., Ferreira, V., Fabbri, R., Oliveira, M.C.F.d., Lopes, A.d.A.: A critical survey of the multilevel method in complex networks. ACM Computing Surveys **53**(2) (2020)
12. Valejo, A., Ferreira, V., de Oliveira, M.C.F., de Andrade Lopes, A.: Community detection in bipartite network: A modified coarsening approach. In: Lossio-Ventura, J.A., Alatrista-Salas, H. (eds.) Information Management and Big Data. pp. 123–136. Communications in Computer and Information Science, Springer International Publishing (2018)
13. Verter, V., Lapierre, S.D.: Location of preventive health care facilities. Annals of Operations Research **110**(1), 123–132 (2002)
14. Walshaw, C.: A multilevel approach to the travelling salesman problem. Oper. Res. **50**(5), 862–877 (2002)
15. Walshaw, C.: Multilevel refinement for combinatorial optimisation problems. Ann. Oper. Res. **131**(1), 325–372 (2004)
16. Zockaie, A., Aashtiani, H.Z., Ghamami, M., (Marco) Nie, Y.: Solving detour-based fuel stations location problems. Computer-Aided Civil and Infrastructure Engineering **31**(2), 132–144 (2016)