Bilevel Large Neighborhood Search for the Electric Autonomous Dial-a-Ride Problem

Steffen Limmer

2023

Preprint:

Transportation Research Interdisciplinary Perspectives 21 (2023) 100876

Contents lists available at ScienceDirect

Transportation Research Interdisciplinary Perspectives

journal homepage: www.elsevier.com/locate/trip

Bilevel large neighborhood search for the electric autonomous dial-a-ride problem

Steffen Limmer

Honda Research Institute Europe GmbH, 63073 Offenbach am Main, Germany

ARTICLE INFO

Keywords:
Dial-a-ride problem
Large neighborhood search
Electric vehicles

ABSTRACT

The electric autonomous dial-a-ride problem (E-ADARP) represents a challenging and practically relevant extension of the dial-a-ride problem, which takes electric vehicle charging into account. It introduces battery constraints and the option to recharge vehicles at different charging stations. The present paper proposes a bilevel large neighborhood search approach (BI-LNS) for the E-ADARP. In the outer level of the proposed approach, charging sessions are inserted in the routes of vehicles and in the inner level, the pick-up and drop-off locations of the requests are inserted. In numerical experiments, it is shown that BI-LNS is able to outperform existing approaches on a number of common E-ADARP benchmark instances. Furthermore, the scalability of BI-LNS is evaluated on a set of large problem instances. The results show that the proposed approach is able to find feasible solutions within five minutes for problem instances with up to a few thousand transportation requests.

1. Introduction

The dial-a-ride problem (DARP) is the problem of finding optimal tours of a number of vehicles through different pick-up and drop-off locations in order to serve a number of transportation requests. Different variants of the DARP, considering different constraints and objectives, can be found in the literature. Most typical constraints are that the routes have to start/stop at certain start/end locations (depots), that the pick-up and drop-off locations have to be served within specific time windows, that a vehicle can transport only a limited number of passengers at the same time, that the user ride time is limited, and that the duration of a vehicle's route is limited. The objective function usually relates to minimizing the operating cost (e.g., in terms of the number of used vehicles) and/or maximizing the user satisfaction (e.g., in terms of user ride times) (Molenbruch et al., 2017b).

The DARP is of high practical relevance due to an increasing popularity of on-demand mobility services (Shaheen and Cohen, 2020). The problem is well studied and numerous approaches for solving the DARP and its variants can be found in the literature. There are different exact methods, which formulate and solve the problem as a mixed integer linear programming (MILP) problem. These methods are mainly based on branch-and-cut approaches (Cordeau, 2006; Ropke et al., 2007; Parragh, 2011; Braekers et al., 2014) and/or column generation approaches (Ropke and Cordeau, 2009; Garaix et al., 2011; Parragh and Schmid, 2013; Gschwind and Irnich, 2015). While such exact methods can be employed to efficiently solve small problem instances and having the advantage of providing guarantees on the solution quality, they are typically not suited for real-world settings, which require low response times and have to handle hundreds or even thousands of requests. Hence, different heuristic solution approaches for the DARP are proposed in the literature. Popular heuristics employed in the context of the DARP are tabu search (Cordeau and Laporte, 2003), large neighborhood search (LNS) (Shaw, 1997; Molenbruch et al., 2017a), adaptive large neighborhood search (Ropke and Pisinger, 2006; Gschwind and Drexl, 2019), and genetic algorithms (Jorgensen et al., 2007). Furthermore, there are several hybrid algorithms for the DARP, which combine two or more approaches (Masmoudi et al., 2016, 2017). For a comprehensive overview to the different variants of the DARP and approaches for their solution, the reader is referred to the excellent literature reviews of Molenbruch et al. (2017b) and of Ho et al. (2018).

Today, the number of electric vehicles (EVs) is rapidly increasing due to economical and especially environmental reasons. At the same time, progress in autonomous driving is made. Hence, it can be assumed that dial-a-ride service fleets of the future consist of autonomous electric vehicles (Narayanan et al., 2020). Since such vehicles do not require a human driver, they can be operated over long time periods without the need for breaks or for returning to a depot. However, in order to be able to provide a continuous service, such vehicles have to be recharged. Since the recharging takes a considerable time, it should be taken into account during the planning of the vehicles' driving schedules. Thus, a practically relevant extension of the DARP is the integration of EV charging, leading to the electric autonomous

E-mail address: steffen.limmer@honda-ri.de.

https://doi.org/10.1016/j.trip.2023.100876
Received 16 May 2022; Received in revised form 27 April 2023; Accepted 4 July 2023
Available online 17 July 2023

Nomenclature

Algorithmic Parameters

C^{init}	Initial number of charging sessions per vehicle
I	Number of inner-level LNS iterations
p_{ins}	Probability for charging session insertion at the outer level
p_{mod}	Probability for charging session modification at the outer level
p_{rem}	Probability for charging session removal at the outer level
Q	Number of requests reinserted at inner level after a feasible solution was found
Q^{in}	Number of requests inserted at inner level as long as no feasible solution was found
Q^{out}	Number of requests removed at inner level as long as no feasible solution was found

Parameters

α_i	Charging speed at charging station i
B	Battery capacity of vehicles
C	Number of charging stations
d_i	Duration of service at location i
$e_{i,j}$	Energy it takes to drive from location i to location j
K	Number of vehicles
L	Load capacity of vehicles
l_i	Load of location i
N	Number of requests
r	Minimum final battery capacity ratio
S	Upper bound for a passenger's ride time
T	Length of planning horizon
$t_{i,j}$	Time it takes to drive from location i to location j
w_i^+	Upper bound of time window of location i
w_i^-	Lower bound of time window of location i

Sets

$\mathcal{C}$	Set of charging stations
$\mathcal{D}+$	Set of start depots
$\mathcal{D}-$	Set of end depots
$\mathcal{K}$	Set of vehicles
$\mathcal{P}+$	Set of pick-up locations
$\mathcal{P}-$	Set of drop-off locations
$\mathcal{V}$	Set of all locations

Variables

A_i	Time of arrival at location i
B_i^{arr}	Vehicle's battery level at arrival at location i
B_i^{dep}	Vehicle's battery level at departure from location i
E_i	Duration of charging at charging station i
L_i	Vehicle's load after serving location i
R_i	Excess ride time for request associated with pick-up location i
S_i	Time of service for location i
$x_{i,j}^k$	Flag indicating whether vehicle k drives from location i to location j
$x_{i,j}$	Flag indicating whether any vehicle drives from location i to location j

dial-a-ride problem (E-ADARP), which was initially introduces by Bongiovanni et al. (2019). In the E-ADARP, it is considered that the vehicles consume energy by driving from one location to another and that it has to be ensured that the vehicles' batteries never get fully discharged. In order to increase the quality of the planned routes, visits at charging stations (CSs) have to be included into them – it has to be planned, when, where and for how long vehicles charge.

This extension makes the already NP-hard DARP even harder to solve, and it cannot be handled by the previously cited approaches for the conventional DARP. The present work proposes a bilevel large neighborhood search approach (BI-LNS) for the E-ADARP. In numerical experiments, the proposed approach is evaluated on common benchmark instances with up to 96 requests and the results are compared to those of the exact e-ADARP2 approach (Bongiovanni et al., 2019) and of the heuristic DA approach (Su et al., 2023). Furthermore, the proposed approach is evaluated on a set of larger problem instances with up to 5200 requests in order to investigate its scalability.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 provides a detailed description of the considered problem. Section 4 describes the proposed BI-LNS approach. In Section 5, the numerical experiments are described and their results are presented and discussed. Finally, a conclusion and summary is provided in Section 6.

2. Related work

In the literature, only very few approaches to the E-ADARP can be found.

Pimenta et al. (2017) propose a hybrid approach for the DARP with electric autonomous vehicles, which operate on fixed routes. However, they do not consider that the vehicles have to be recharged. Venkatraman and Levin (2019) assume that private vehicles are replaced by a fleet of autonomous vehicles, leading to traffic congestion, and propose a tabu search approach for the planning of the operation of the vehicles taking into account the traffic congestion. A similar problem is considered by Liang et al. (2020), who propose an exact approach for its solution.

Masmoudi et al. (2018) investigated the DARP with EVs, which can swap their batteries at battery swapping stations. Thus, in the considered problem, it is not necessary to set charging durations. The authors propose a hybrid approach, which combines variable neighborhood search with evolutionary optimization, and evaluate it on problem instances with up to 100 requests.

Bongiovanni et al. (2019) proposed an exact approach for the E-ADARP. They provide a 3-index (e-ADARP3) and a more efficient 2-index (e-ADARP2) MILP formulation of the E-ADARP and propose a branch-and-cut approach with new valid inequalities specific to the E-ADARP. They evaluate the approach on problem instances with up to 50 requests.

In (Bongiovanni, 2020), Bongiovanni proposes a two-phase heuristic for the dynamic version of the E-ADARP. In the first phase, it is tried to insert newly received requests into existing routes with help of a greedy operator, which appends a new charging session at the end of routes into which new requests were inserted. In the second phase, a large neighborhood search is applied in order to improve the routes.

Table 1
Overview to related and present work.

Work	Approach	Largest considered problem size	Considered problem variant
(Masmoudi et al., 2018)	Population-based variable neighborhood search	17 EVs, 100 requests	Battery swapping instead of charging, all requests have to be satisfied, maximum ride time constraint
(Bongiovanni et al., 2019)	Branch-and-cut (e-ADARP2 and e-ADARP3)	5 EVs, 50 requests	Same variant as in present work
(Hoché et al., 2020)	Neighborhood search	35 EVs, 10,000 requests	No hard constraint for satisfaction of requests, constraint that a charging station cannot be used by two EVs simultaneously, no maximum ride time constraint
(Bongiovanni, 2020)	Greedy heuristic followed by large neighborhood search supported by machine learning approach	10 EVs, number of requests not specified	Dynamic variant, which consists of solving multiple subproblems, no hard constraint for satisfaction of newly received requests, maximum ride time constraint
(Su et al., 2023)	Deterministic annealing (DA)	8 EVs, 96 requests	Same variant as in present work
Present work	Bilevel large neighborhood search	250 EVs, 5200 requests	Static variant, all requests have to be served, maximum ride-time constraint

The destroy and repair operators in the LNS are chosen with help of a machine learning model, which is trained in an offline stage. Furthermore, Bongiovanni proposes an exact algorithm for the setting of service times of existing routes, which minimizes the excess ride time (the time passengers spend in vehicles longer than necessary), and shows that the algorithm is faster than a linear programming approach.

Hoché et al. (2020) proposed a heuristic neighborhood search approach. It uses a special insertion operator in order to take the EV charging into account: A request is first inserted into a route of a vehicle without consideration of battery constraints and then, the idle times of the vehicle (i.e., times, when the vehicle is waiting at a location for the start of the time window) are filled with visits to charging stations. If the resulting route is feasible, it is accepted and otherwise the insertion of the request failed. The authors evaluate the approach on problem instances with 10,000 requests and 35 vehicles. However, only limited details to the experimental results are provided.

A further heuristic approach for the E-ADARP is proposed by Su et al. (2023). They describe a deterministic annealing approach (DA), which employs different local search operators. A special repair operator is used in order to insert visits at charging stations into routes that violate battery constrains. The approach is evaluated on problem instances with up to 96 requests, and it is shown that on several instances, it is able to outperform the exact e-ADARP2 approach.

The heuristic approaches proposed by Su et al. (2023) and by Hoché et al. (2020), both repair battery-infeasible routes by inserting charging sessions in the existing routes. The in this work proposed approach for the E-ADARP works in the opposite way: Instead of inserting requests and then filling up the route with charging sessions, the charging sessions are first inserted, and then the route is filled up with requests. This is done in the form of a bilevel large neighborhood search approach (BI-LNS), where the outer level is responsible for setting the visits to charging stations and the inner level is responsible for filling up the routes with requests. By iteratively modifying the charging sessions at the outer level, the number of charging sessions and their settings (locations and durations) are improved over time. The approach does not employ a compute-intensive operator for the insertion of charging sessions, which can be expected to be beneficial for the scalability. Table 1 provides a summary of the most relevant related work and the present work.

The key contributions of the present paper are as follows:

- Proposal of a new bilevel large neighborhood search heuristic for the E-ADARP,
- Presentation of new results on E-ADARP instances known from literature, and
- Introduction of a set of large-scale E-ADARP instances and evaluation of the scalability of the proposed approach on these instances.

3. Problem description

A problem analogous to that described by Bongiovanni et al. (2019) is considered. It is assumed that a set $\mathcal{K} = \{1, \dots, K\}$ of K electric vehicles is available to serve N requests. For each request, there is a pick-up and a drop-off location. Let $\mathcal{P}^+ = \{1, \dots, N\}$ denote the set of pick-up locations and $\mathcal{P}^- = \{N+1, \dots, 2N\}$ denote the set of drop-off locations, where $i \in \mathcal{P}^+$ and $i + N \in \mathcal{P}^-$ are the pick-up and drop-off location, respectively, belonging to the ith request. Each vehicle is associated with a start and an end depot. Let $\mathcal{D}^+ = \{2N+1, \dots, 2N+K\}$ be the set of start depots and $\mathcal{D}^- = \{2N+K+1, \dots, 2N+2K\}$ be the set of end depots. Furthermore, there is a set $C = \{2N+2K+1, \dots, 2N+2K+C\}$ of C charging stations. Optionally, if it is permitted to visit a physical charging station more than once, the set C might contain a sufficient number of "virtual charging stations" or "charging sessions", which are replications of the physical charging station locations. Let $\mathcal{V} = \mathcal{P}^+ \cup \mathcal{P}^- \cup \mathcal{D}^+ \cup \mathcal{D}^- \cup C$ denote the set of all locations of the problem. The ride from a location i to a location j, takes $t_{i,j}$ time units and consumes $e_{i,j}$ energy units. Each vehicle has a maximum battery capacity of B energy units and a maximum load capacity L, where the latter represents the maximum number of passengers, a vehicle can transport at the same time. Without loss of generality, it is assumed that at the beginning of the planning horizon, the vehicles are empty and have fully charged batteries. Each location $i \in \mathcal{V}$ is associated with a load (number of passengers) l_i with $l_i = 0$ for all locations $i \in \mathcal{V} \setminus (\mathcal{P}^+ \cup \mathcal{P}^-)$ and $l_i > 0$ for all pick-up locations $i \in \mathcal{P}^+$, and $l_i = -l_{i-N}$ for all drop-off locations $i \in \mathcal{P}^-$. For each location $i \in \mathcal{V}$, a time window $[w_i^-, w_i^+]$ is defined. For all locations $i \in \mathcal{V} \setminus (\mathcal{P}^+ \cup \mathcal{P}^-)$, w_i^- is set to 0 and w_i^+ is set to the length T of the planning horizon. It is assumed that the service (i.e., the pick-up and drop-off of passengers) at location i takes a certain time d_i with $d_i = 0$ for locations $i \in \mathcal{V} \setminus (\mathcal{P}^+ \cup \mathcal{P}^-)$. Each charging station $i \in C$ has a certain charging speed of α_i energy units per time unit. Furthermore, it is assumed that the battery level of a vehicle at the end of the planning horizon must not be lower than a ratio $r \in [0, 1]$ of its maximum battery capacity and that the time a passenger spends in a vehicle is limited by a maximum ride time S.

A solution to the problem consists of routes of the vehicles, i.e., a sequence of locations for each vehicle, together with service times S_i (i.e., points in time when locations are served) for each visited location $i \in \mathcal{V}$. Charging durations directly follow from the service times. Let the binary variable $x_{i,j}^k$ indicate whether vehicle k visits location j directly after location i in some given routes. An objective function and constraints as described in the following two subsections are assumed.

3.1. Objective

The goal is to find feasible routes, which are optimal with respect to the following objective:

$$\min w_1 \sum_{i,j \in \mathcal{V}} \sum_{k \in \mathcal{K}} t_{ij} x_{i,j}^k + w_2 \sum_{i \in \mathcal{P}^+} R_i, \tag{1}$$

where w_1 and w_2 are prespecified weights and R_i is the excess ride time of request i, defined as follows:

$$R_i = S_{i+N} - (S_i + d_i) - t_{i,i+N} \qquad \forall i \in \mathcal{P}^+. \tag{2}$$

Thus, the excess ride time is the time passengers spend in a vehicle longer than necessary and the total objective is to reduce (a weighted sum of) the total driving time of the vehicles and the excess ride time of passengers.

3.2. Constraints

Routes have to fulfill the following constraints in order to be feasible.

Each location can be visited at most once (Constraint (3)). A vehicle has to start its route at its corresponding start depot and cannot approach any start depot (constraints (4) and (5)). Furthermore, each vehicle's route has to end at its corresponding end depot (Constraint (6)) and if a vehicle enters a location other than an end depot, it also has to leave the location (Constraint (7)):

$$\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{V}} x^k_{i,j} \le 1 \qquad \forall j \in \mathcal{V}, \tag{3}$$

$$\sum_{j\in\mathcal{V}\setminus\{2N+k\}} x^k_{2N+k,j} = 1 \qquad \forall k \in \mathcal{K}, \tag{4}$$

$$\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{V}} x^k_{i,j} = 0 \qquad \forall j \in \mathcal{D}^+, \tag{5}$$

$$\sum_{i\in\mathcal{V}\setminus\{2N+K+k\}} x^k_{i,2N+K+k} = 1 \qquad \forall k \in \mathcal{K}, \tag{6}$$

$$\sum_{j\in\mathcal{V}} x^k_{j,i} - \sum_{j\in\mathcal{V}} x^k_{i,j} = 0 \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{V}\setminus\mathcal{D}^-. \tag{7}$$

Each pick-up location has to be visited by exactly one vehicle (Constraint (8)) and a vehicle visiting a pick-up, has to visit also the corresponding drop-off (Constraint (9)).

$$\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{V}} x^k_{i,j} = 1 \qquad \forall j \in \mathcal{P}^+, \tag{8}$$

$$\sum_{i\in\mathcal{V}} x^k_{i,j} = \sum_{i\in\mathcal{V}} x^k_{i,j+N} \qquad \forall k \in \mathcal{K}, \forall j \in \mathcal{P}^+. \tag{9}$$

Note that this means that a solution, in which not all requests are served, is infeasible.

In the following, the binary flag $x_{i,j}$ is used to indicate whether any of the vehicles drives from location i to location j:

$$x_{i,j} = \sum_{k\in\mathcal{K}} x^k_{i,j} \qquad \forall i,j \in \mathcal{V}. \tag{10}$$

The service time S_i of a location $i \in \mathcal{V}$ must not be earlier than the corresponding arrival time A_i (set in (11) and (12)) and has to adhere to the time window constraint (constraints (13) and (14)). Furthermore, a drop-off location cannot be visited/served earlier than the corresponding pick-up location (Constraint (15)), and the maximum ride time constraint has to be satisfied (Constraint (16)):

$$A_i = 0 \qquad \forall i \in \mathcal{D}^+, \tag{11}$$

$$A_i = \sum_{j\in\mathcal{V}} (S_j + d_j + t_{j,i}) \cdot x_{j,i} \qquad \forall i \in \mathcal{V}\setminus\mathcal{D}^+, \tag{12}$$

$$S_i \ge A_i \qquad \forall i \in \mathcal{V}, \tag{13}$$

$$w_i^- \le S_i \le w_i^+ \qquad \forall i \in \mathcal{V}, \tag{14}$$

$$S_i \le S_{i+N} \qquad \forall i \in \mathcal{P}^+, \tag{15}$$

$$S_{i+N} - (S_i + d_i) \le S \qquad \forall i \in \mathcal{P}^+. \tag{16}$$

The load L_i after visiting location i (set in (17) and (18)) cannot be larger than the maximum load capacity L (Constraint (19)) and charging stations can only be visited by empty vehicles (Constraint (20)):

$$L_i = 0 \qquad \forall i \in \mathcal{D}^+, \tag{17}$$

$$L_i = l_i + \sum_{j\in\mathcal{V}} L_j \cdot x_{j,i} \qquad \forall i \in \mathcal{V}\setminus\mathcal{D}^+, \tag{18}$$

$$L_i \le L \qquad \forall i \in \mathcal{V}, \tag{19}$$

$$L_i = 0 \qquad \forall i \in \mathcal{C}. \tag{20}$$

The vehicles start their tours with full batteries (Constraint (21)). The battery level B_i^{arr} at arrival at a location (set in (22)) cannot fall below zero (Constraint (23)) and the final minimum battery level ratio has to be satisfied (Constraint (24)). It is assumed that at a charging station $i \in \mathcal{C}$, the time between arrival and service is used for charging and thus, the battery level B_i^{dep} at departure from the charging station is up to $(S_i - A_i) \cdot \alpha_i$ energy units higher than at arrival (constraints (25) and (26)). For all other locations, the battery level does not change between arrival and departure (Constraint (27)):

$$B_i^{arr} = B \qquad \forall i \in \mathcal{D}^+, \tag{21}$$

$$B_i^{arr} = \sum_{j\in\mathcal{V}} (B_j^{dep} - e_{j,i}) \cdot x_{j,i} \qquad \forall i \in \mathcal{V}\setminus\mathcal{D}^+, \tag{22}$$

$$B_i^{arr} \ge 0 \qquad \forall i \in \mathcal{V}, \tag{23}$$

$$B_i^{arr} \ge r \cdot B \qquad \forall i \in \mathcal{D}^-, \tag{24}$$

$$E_i = S_i - A_i \qquad \forall i \in \mathcal{C}, \tag{25}$$

$$B_i^{dep} = \min(B, B_i^{arr} + E_i \cdot \alpha_i) \qquad \forall i \in \mathcal{C}, \tag{26}$$

$$B_i^{dep} = B_i^{arr} \qquad \forall i \in \mathcal{V}\setminus\mathcal{C}. \tag{27}$$

Please note that in contrast to the problem definition of Bongiovanni et al. (2019), it is not considered that vehicles can select from multiple end depots. However, on the benchmark instances used in the experiments described later, this makes no difference, since there is only a single end depot in these instances.

4. BI-LNS approach

This work proposes a bilevel large neighborhood search approach (BI-LNS) for solving the stated problem. Large neighborhood search (LNS) is an established local search heuristic for solving the DARP. LNS iteratively destroys and repairs solutions. In the context of the DARP, this means that requests are iteratively removed and reinserted from/into routes. In the literature, different operators for the removal and insertion of requests are proposed. Examples of popular removal and insertion operators are *random removal*, *worst removal*, *greedy insertion*, *deep greedy insertion*, and *regret insertion* (Ropke and Pisinger, 2006; Vallée et al., 2017). A popular extension of LNS is adaptive large neighborhood search (Ropke and Pisinger, 2006), which in each iteration dynamically chooses from a set of different removal and insertion operators.

In order to be able to schedule the visits to charging stations in the E-ADARP, LNS is extended to a bilevel approach. The outer level (Section 4.1) sets and modifies charging sessions of vehicles and removes requests from affected routes. The inner level (Section 4.2) inserts requests with help of a greedy insertion operator (Section 4.3) under consideration of the charging sessions set by the outer level. Thus, at the outer level charging sessions are fixed and at the inner level, the

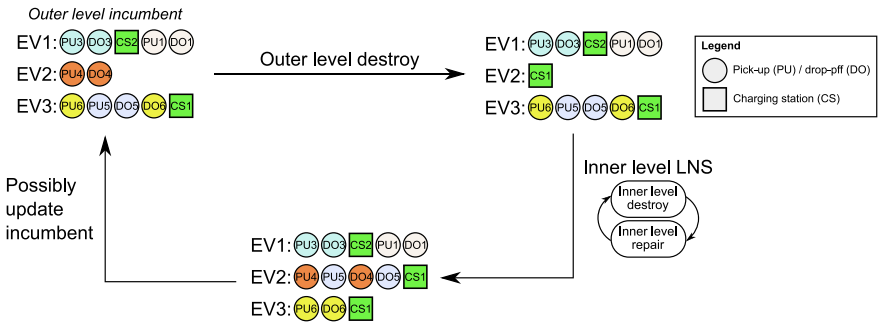

Fig. 1. Illustration of proposed approach for planning routes of three EVs.

requests are arranged around the charging sessions. It can be expected that the initial setting of the charging sessions is far from being optimal. However, iteratively the charging sessions are improved by applying modifications to them. Since these modifications are small/local, the inner level does not have to start from scratch in each iteration of the outer level. Fig. 1 exemplary illustrates the proposed approach for planning routes of three EVs. In the shown example, the outer level modifies the route of the second EV.

One might argue that the proposed approach wastes too much time of the search with arranging requests around suboptimal charging sessions. An alternative approach would be to combine the (re-)insertion of requests with the setting of charging sessions. That means that the charging sessions are not fixed before the insertion of requests, but that in the case that an insertion makes a route battery infeasible, it is tried to make the route feasible by modifying/inserting charging sessions in the existing route. However, determining a charging session setting, which makes the route feasible is time consuming. Thus, this alternative approach wastes much time with setting/inserting charging sessions in routes with a suboptimal setting of requests. Furthermore, with an increasing number of requests in a route, the options for modifying/inserting charging sessions in the route decreases. This makes the alternative approach prone to getting stuck in a local optimum. The proposed bilevel approach can be expected to be beneficial in terms of a better exploration of the search space.

In the following, the main components of the approach are described in detail. In the explanation, it is assumed that in contrast to the previous problem formulation, the charging stations can be visited arbitrarily often. In Section 4.4, it is explained, how the approach can be adapted if each charging station can be visited only once.

4.1. Outer-level LNS

The outer-level LNS is responsible for setting the visits to charging stations in the routes. Algorithm 1 outlines how the outer-level LNS works. It starts with empty routes, in which all vehicles directly drive from their start depot to their end depot. It then inserts in each vehicle's route a prespecified number C^{init} of randomly generated charging sessions. Each charging session c has the following parameters:

- a location $c.i \in C$ of the physical charging station to charge at,
- a charging duration $c.d$, and
- a time window $c.w = [c.l, c.u]$ for the service time.

The physical charging station $c.i$ is uniformly sampled from the set C of available charging stations. The charging duration $c.d$ is uniformly random chosen between zero and the time $d^{max} = B/\alpha_{c.i}$ it takes to fully charge a vehicle's battery at the selected charging station. The lower bound $c.l$ of the time window is set to zero and the upper bound $c.u$ is uniformly random chosen from the interval $[t_{2N+k,c.i}, T - t_{c.i,2N+K+k}]$, where k is the vehicle in whose route the charging session is inserted. In each route, the charging sessions are inserted between the start and end depot in increasing order of their time window upper bound $c.u$.

Algorithm 1: Outer-level LNS.

Input: *requests*
Output: routes rt^*
Parameter: C^{init}, I, p_{mod}, p_{rem}

```
rt* = empty_routes();
rt* = insert_random_cs(rt*, C^init);
rt*, rej_req* = inner_LNS(rt*, requests, I);
o* = obj(rt*);
while termination condition not met do
    rt = rt*;
    rej_req = rej_req*;
    p = rand(0,1);
    if p ≤ p_mod and at least one c in rt then
        modify random c in rt;
        V = vehicle in whose route c was modified;
    else if p ≤ p_mod + p_rem and at least one c in rt then
        remove random c from rt;
        V = vehicle from whose route c was removed;
    else
        insert random c in rt;
        V = vehicle in whose route c was inserted;
    end
    rt[V], req = remove_requests(r[V]);
    rej_req = rej_req ∪ req;
    rt, rej_req = inner_LNS(rt, rej_req, I);
    o = obj(rt);
    if (|rej_req*| > 0 and |rej_req| ≤ |rej_req*|) or
      (|rej_req*| == |rej_req| == 0 and o ≤ o*) then
        rt* = rt;
        o* = o;
        rej_req* = rej_req;
    end
end
return rt*;
```

After initializing the routes, the inner-level LNS is then run for a prespecified number I of iterations in order to insert the requests (i.e., the corresponding pick-up and drop-off locations) in the routes. The resulting routes are stored as currently best routes rt^*. They might violate the constraint that all requests have to be served, because the inner-level LNS was possibly not able to insert all requests without violating time, load or battery constraints. The requests, which could not be inserted are stored as the currently best set of rejected requests rej_req^* and the objective value of rt^* is stored as the currently best objective value o^*.

In order to improve the currently best routes (i.e., increase the number of served requests and/or improve the objective function value), the following procedure is repeated until a termination condition (e.g., a maximum number of iterations, a time limit, or a threshold for the

solution quality) is met: A copy rt of the currently best routes rt^* and a copy rej_req of the currently best set of rejected requests rej_req^* is made and the charging sessions in rt are varied by one of the following variation operators:

- Charging session modification mod,
- Charging session insertion ins, or
- Charging session removal rem.

The mod operator randomly selects a charging session from rt and removes it together with all pick-up and drop-off locations from the corresponding route. Then, it inserts a new random charging session in the route at a position which ensures that the charging sessions in the route are still sorted in increasing order of their upper time window bounds. The ins operator randomly selects a route, removes the pick-up and drop-off locations from it and inserts a new random charging session in the route (preserving the correct order of the charging sessions). The rem operator works like the mod operator without insertion of a new charging session. If rt does not contain any charging session, the ins operator is selected as variation operator. Otherwise, the used operator is randomly selected with different probabilities p_{mod}, p_{ins}, p_{rem}, respectively, for the different operators. The requests, which were removed from rt by the variation operator, are added to the set of rejected requests rej_req.

Then, I iterations of the inner-level LNS are executed on rt in order to try to insert the requests rej_req. If the resulting routes rt serve at least as many requests as rt^* and one of the following conditions holds:

- rt^* does not serve all requests, or
- rt^* serves all requests and the objective value o of rt is less or equal the objective value o^* of rt^*,

then rt is stored as new currently best routes rt^* and otherwise it is rejected.

4.2. Inner-level LNS

The inner-level LNS works as outlined in Algorithm 2. It takes existing routes and a set of requests to insert as input and first tries to insert the requests with help of an insertion operator. Then it tries to improve the resulting routes rt^* by iteratively (re-)inserting requests. In each iteration, it removes Q^{out} randomly selected requests from the currently best routes and tries to insert Q^{in} randomly selected unserved requests with help of the insertion operator. Similar to the outer-level LNS, if the resulting routes are at least as good as the currently best routes in terms of the number of served requests and/or the objective value, they are accepted as new currently best routes and otherwise they are rejected.

As long as not all requests are served in the currently best routes rt^*, Q^{in} is set to a larger value than Q^{out} in order to try to increase the number of served requests. After a solution, which serves all requests, is found, Q^{in} and Q^{out} are both set to the same value Q.

4.3. Insertion operator

For the actual insertion of requests, a greedy operator is used. Its working is outlined in Algorithm 4 in Appendix A. It takes as input existing routes and a series of requests to insert and tries to insert each of these requests in the given order. For each request, it traverses the current routes and determines all combinations of positions where the corresponding pick-up and drop-off locations can be inserted without violating constraints. If at least one such combination exists, the pick-up and drop-off locations are inserted at the positions of the combination which results in the lowest increase of the objective function value and otherwise, the request is not inserted. In order to reduce the risk of getting stuck in a local optimum, random noise is added to the increases of the objective function values associated to the different position combinations, as proposed by Ropke and Pisinger (2006).

Algorithm 2: Inner-level LNS.

Input: routes rt, $new\_requests$, I
Output: routes rt^*, rejected requests rej_req^*
Parameter: Q^{in}, Q^{out}, Q

1 rt^*, rej_req^* = insertion_operator(rt, $new\_requests$);
2 o^* = obj(rt^*);
3 i = 0;
4 **while** $i < I$ **do**
5 **if** $|rej\_req^*| == 0$ **then**
6 $Q^{in} = Q$;
7 $Q^{out} = Q$;
8 **end**
9 $rt = rt^*$;
10 $rej\_req = rej\_req^*$;
11 rt, req = remove_requests(rt, Q^{out});
12 $rej\_req = rej\_req \cup req$;
13 req = select_requests_to_insert(rej_req, Q^{in});
14 rt, rej_req = insertion_operator(rt, req);
15 o = obj(rt);
16 **if** $(|rej\_req^*| > 0$ ***and*** $|rej\_req| \leq |rej\_req^*|)$ ***or*** $(|rej\_req^*| == |rej\_req| == 0$ ***and*** $o \leq o^*)$ **then**
17 $rt^* = rt$;
18 $o^* = o$;
19 $rej\_req^* = rej\_req$;
20 **end**
21 i++;
22 **end**
23 **return** rt^*, rej_req^*;

When checking the feasibility of a route (lines 12 and 15 in Algorithm 4), charging sessions are taken into account as follows: For each charging session c in the route, it is assumed that the charging takes place at position $c.i$ with a charging speed of $\alpha_{c.i}$. It is checked that the time window $c.w$ is satisfied and that the station is not visited with a passenger on board. It is assumed that the vehicle charges for $c.d$ time units after arriving at the station and thus, the service time of the charging session is the arrival time plus $c.d$.

Without the maximum ride time constraint and the excess ride time in the objective function, it would be sufficient to set the service times of all other locations to their corresponding arrival times. However, with the given problem formulation it might be beneficial to reduce the passenger ride times by delaying the services at pick-up locations. In order to check whether a given route, which satisfies the time window constraint, also satisfies the ride time constraint, the 8-step approach based on forward time slacks as described by Cordeau and Laporte (2003) is used. It shifts the services at the pick-up locations to the latest possible points in time without violation of the time window constraint and without an increase in the violation of the ride time constraint. Then it checks whether the ride time constraint is satisfied with the given service times. The same approach is used to delay the service times before computing the objective function value. It has to be noted that while this approach typically reduces the excess ride time compared to setting the service times at pick-ups to the earliest possible points in time, it cannot guarantee an optimal excess ride time.

4.4. Adaption to handle limited charging station visits

If a charging station $i \in C$ can be visited only once, the outer-level LNS, described in Section 4.1, has to be adapted. The following approach is used to deal with the constraint of limited CS visits: The initialization of the routes with charging sessions (insert_random_cs in line 2 of Algorithm 1) is done as outlined in Algorithm 3. Lines 5 and 7 ensure that a different charging station location is chosen in each

Algorithm 3: Initialization of routes with charging sessions under consideration of constraint of limited charging station visits.

```
   Input: routes, C^init
   Output: routes
1  n = 0;
2  for each vehicle k ∈ K do
3      for j = 1, ... C^init do
4          if n < C then
5              n = n + 1;
6              c = new_charging_session();
7              c.i = 2N + 2K + n;
               // Randomly initialize remaining
                  parameters of c
8              c = init(c);
9              insert c in routes[k];
10         end
11     end
12 end
13 return routes;
```

iteration of the inner loop and if all available locations are already in use, the condition in the if-statement in line 4 evaluates to false and no further charging sessions are initialized/inserted. Besides the insertion operator, the variation operators are adapted as follows: The *mod* variation operator inserts a new charging session with the same physical charging station $i \in C$ as the previously removed charging session. The *ins* operator takes care that the physical charging station of the inserted charging session is not already in the routes. If all physical charging stations are used in the routes, the *mod* operator is used instead of the *ins* operator.

5. Numerical experiments

All experiments with BI-LNS are executed on a 3.8 GHz Intel Core i5-7600K quad-core CPU with 15.6 GB RAM using a single-threaded C/C++ implementation of BI-LNS. In the inner-level LNS of BI-LNS, Q^{in} and Q^{out} (the number of inserted and removed requests, respectively, before a feasible solution is found) are set to two and one, respectively. The requests are initially sorted in increasing order of the lower bound of their pick-up time windows. That means, the greedy insertion operator processes the requests in this order, when it is called for the first time. In the greedy insertion operator, a random noise between −1% and +1% is added to the objective function values. The setting of further parameters of BI-LNS is discussed in Section 5.2.

5.1. Problem instances

In order to evaluate the proposed BI-LNS approach, experiments are executed on three sets of benchmark instances.[1][2] The first set consists of extensions of 14 problem instances introduced by Cordeau (2006) with up to five vehicles and 50 requests. The pick-up and drop-off locations are set in the square $[-10, -10] \times [10, 10]$ and the travel times between locations in minutes equal the Euclidean distances between the locations. All start and end depots are located at the center of the square. The maximum ride time S is set to 30 min. Each vehicle has a load capacity of three passengers and there is one passenger per request. The service durations are set to three minutes for all pick-up and drop-off locations. The second set of benchmark instances consists of extensions of 10 instances introduced by Ropke et al. (2007). They are generated similar to the Cordeau instances but contain up to 8 vehicles and 96 requests. The third benchmark instance set consists of five large instances with up to 260 vehicles and 5200 requests, which were generated by ourselves similar to the Cordeau and Ropke instances. In the Cordeau and Ropke instances, the largest planning horizon T is 720 min and there are up to 12 requests per vehicle. In the large instances, the planning horizon is set to 1200 min and there are 20 requests per vehicle. The problem instances of all three sets are extended as described by Bongiovanni et al. (2019): Three charging stations are added: One at the center of the square, one at the position $(-4, -4)$, and one at the position $(4, 4)$. It is assumed that a vehicle consumes and charges 0.055 kWh of energy per minute of ride and charging time, respectively. The maximum battery capacity B is set to 14.85 kWh. The weights in the objective function (1) are set to $w_1 = 0.75$ and $w_2 = 0.25$. The naming scheme of all benchmark instances is a$K - N$, where K is the number of vehicles and N is the number of requests considered in the instance.

5.2. Parameter tuning and sensitivity analysis

In this section, the setting of the BI-LNS parameters I (number of inner-level LNS iterations), Q (number of reinsertions per iteration of the inner-level LNS after a feasible solution was found), p_{mod}, p_{ins}, and p_{rem} (probabilities of the variation operators at the outer level) is discussed. In order to determine reasonable settings of the parameters, ten tuning problem instances are generated, on which different parameter settings are evaluated. The tuning instances are generated analogously to the problem instances discussed in Section 5.1. For each instance, the number K of vehicles is chosen randomly between 2 and 8, the number of requests per vehicle is chosen randomly from $\{8, 10, 12\}$, the minimum final battery capacity ratio r is chosen randomly from $[0.35, 0.75)$, and the planning horizon T is set to 720 min. The characteristics of the resulting tuning instances are provided in Appendix C.

It can be considered reasonable to set the probability p_{ins} of insertion of a charging session equal to the probability p_{rem} of removal of a charging session. Thus, only the probability p_{mod} of modification of a charging session is tuned and the remaining probabilities are set to $p_{ins} = p_{rem} = (1 - p_{mod})/2$. In order to determine a good starting point for a sensitivity analysis, 100 iterations of random search are executed. In each iteration, a random parameter combination is generated, where I is chosen from $\{1000, \dots, 20000\}$, Q is chosen from $\{2, \dots, 20\}$, and p_{mod} is chosen from $[0.5, 1)$. Then, BI-LNS is run with the resulting parameter combination on the ten tuning instances. Per instance, a time limit of 5 min is set. Furthermore, no limit for the number of visits of a physical charging station is set, and the initial number C^{init} of charging sessions per vehicle to is set to 1. With all of the 100 evaluated parameter combinations, BI-LNS was able to find feasible solutions for all tuning instances. Fig. 2 shows the evaluated parameter combinations and the resulting mean objective values over the 10 tuning instances. The best parameter combinations found are located in an area with low values for I and high values for Q. The combination with $I = 1792$, $Q = 19$, and $p_{mod} = 0.54$ yielded the best mean objective of 556.27. Thus, the parameter setting $I = 1800$, $Q = 19$, $p_{mod} = 0.55$ is used as default setting for a sensitivity analysis.

In the sensitivity analysis, one parameter is varied at a time while keeping the other parameters to their default values. Again, each of the resulting parameter combinations is evaluated on the tuning instances with a time limit of 5 min per instance, with $C^{init} = 1$, and with unlimited visits of charging stations. The results are shown in Tables 2, 3, and 4. One can see that beginning with a value of around 4800, the results deteriorate with an increasing value if I. For Q and p_{mod}, no clear trend can be observed. Thus, the results indicate that the sensitivity of the BI-LNS regarding the setting of these parameters is

[1] The extended Cordeau instances were made publicly available by Bongiovanni et al. under https://luts.epfl.ch/wpcontent/uploads/2019/03/e_ADARP_archive.zip.

[2] The extended Ropke instances and large instances are available under https://github.com/HRI-EU/e_adarp_material.

Table 2
Mean objective over the ten tuning instances resulting from different settings of I with $Q = 19$ and $p_{mod} = 0.55$.

I	800	1800	2800	3800	4800	5800	6800	7800	8800	9800	10,800
Mean Obj	554.21	559.97	559.29	558.23	561.11	560.58	561.21	562.40	564.69	564.89	563.23

Table 3
Mean objective over the ten tuning instances resulting from different settings of Q with $I = 1800$ and $p_{mod} = 0.55$.

Q	5	7	9	11	13	15	17	19
Mean Obj	559.82	556.89	557.19	557.70	558.54	557.43	558.60	559.97

Table 4
Mean objective over the ten tuning instances resulting from different settings of p_{mod} with $I = 1800$ and $Q = 19$.

p_{mod}	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
Mean Obj	557.96	557.49	559.97	557.02	556.55	555.92	557.16	558.95	557.63	555.74	556.25

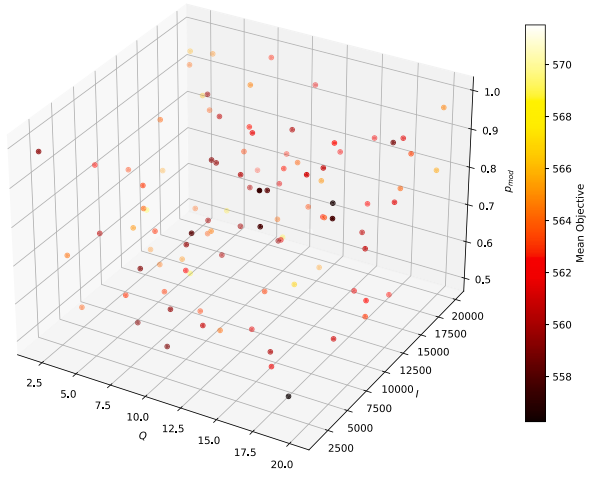

Fig. 2. 100 parameter combinations evaluated by random search and the corresponding mean objective over the tuning instances.

low. The best mean objective was obtained with the parameter setting $I = 800$, $Q = 19$, and $p_{mod} = 0.55$. Thus, this setting is used for BI-LNS in the experiments on the benchmark instances described in the following sections.

5.3. Experimental setup

On the Cordeau and Ropke instances, the optimizations with BI-LNS are run with two settings: In the first setting, it is assumed that each physical charging station can be visited only once and the adaption described in Section 4.4 is used. In the second setting, it is assumed that the number of visits of physical charging stations is not limited. On the large instances, only the second setting is used. For each used setting and each problem instance, 10 trials with 10 different seeds are executed. If not otherwise stated, a time limit of five minutes is set per trial. The initial number C^{init} of charging sessions per vehicle is set to one on the Cordeau and Ropke instances. On the large instances, it is set to two since the planning horizon is larger and there are more requests per vehicle.

The results of BI-LNS are compared with the results of the exact e-ADARP2 approach as reported by Bongiovanni et al. (2019) and with the result of the DA heuristics as reported by Su et al. (2023). Bongiovanni et al. (2019) executed the experiments with the exact e-ADARP2 approach using version 7.0.1 of the Gurobi solver with a time limit of 120 min per problem instance on a 3.6 GHz Intel Core CPU with 16 GB RAM. The results of the DA approach were computed by Su et al. (2023) with a Julia 1.7.2 implementation with 50 trials per problem instance and 10,000 iterations per trial on a 2.1 GHz Intel Xeon Gold 6230 Core CPU.

5.4. Experimental results

5.4.1. Results on Cordeau instances

Tables 5 and 6 show the results of e-ADARP2, DA, and BI-LNS on the Cordeau instances with allowing only up to one visit to each charging station. In Table 5, the results are shown for minimum final battery capacity ratios of $r = 0.1$ and $r = 0.4$, and Table 6 shows the results for $r = 0.7$. For the e-ADARP2 approach, the runtimes (RT) and final objective values are shown, for DA, the average runtimes and minimum and average objective values over the 50 trials are shown, and for BI-LNS, the minimum, mean, and maximum objective values over all trials, which yielded a feasible solution, are shown. Furthermore, for BI-LNS it is shown how many of the 10 trials yielded a feasible solution. The runtime of the BI-LNS approach is not shown since it is always five minutes per trial. Su et al. (2023) do not provide a mean objective value for DA on problem instances where not all of the 50 trials yielded a feasible solution. "NA" means that no feasible solution was found and a "*" indicates that the result is proven optimal according to Bongiovanni et al. (2019). As already described in (Su et al., 2023), for some problem instances, the global optimum reported in (Bongiovanni et al., 2019) is in fact not globally optimal. These results of e-ADARP2 are marked with "**" in Tables 5 and 6. For most of the corresponding problem instances, we were able to compute the actual global optimum with an exact MILP formulation (see Appendix B for more details). The corrected global optima are shown in brackets behind the results of e-ADARP2.

As one can see, with $r = 0.1$ and $r = 0.4$, e-ADARP2 is very efficient. DA is also able to find solutions of high quality. Both e-ADARP2 and DA outperform BI-LNS on most problem instances. However, the gaps between BI-LNS and e-ADARP are limited. For $r = 0.1$, the highest average gap is 0.81% (on a3-18) and the highest worst gap is 0.94% (on a3-36). For $r = 0.4$, the highest average gap is 0.99% (on a4-48) and the highest worst gap is 1.62% (on a4-48). With $r = 0.7$, the situation changes compared to $r = 0.1$ and $r = 0.4$. The e-ADARP2 approach is no longer able to find feasible solutions for all problem instances within 120 min. On 10 of the 14 Cordeau instances, DA finds feasible solutions in all trials, while BI-LNS finds feasible solutions in all trials on 12 instances. On the smaller instances, BI-LNS is still outperformed by DA and e-ADARP2, but on the larger problem instances, it is able find the best solutions. A reason why BI-LNS is outperformed by the other two approaches on smaller instances and instances with a low value for r might be that e-ADARP2 and DA are better in fine-tuning solutions, especially in terms of service times and charging durations. Thus, they have an advantage in exploitation (finding local optima within promising regions of the search space), while BI-LNS can be expected to have an advantage in exploration (finding promising regions of the search space). On easier instances, the exploitation is more important, while on the harder instances, BI-LNS can benefit from the better exploration.

For the case that multiple visits to a charging station are allowed, no results of e-ADARP2 are provided in (Bongiovanni et al., 2019). With

Table 5
Results on Cordeau instances with limited CS visits and $r = 0.1$ and $r = 0.4$.

Instance	e-ADARP2 (Bongiovanni et al., 2019)		DA (Su et al., 2023)			BI-LNS			Feasible
	RT [min]	Obj	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Obj max	
				r = 0.1					
a2–16	0.02	**237.38***	0.65	**237.38***	**237.38***	238.20	238.20	238.20	10/10
a2–20	0.07	**279.08***	1.23	**279.08***	**279.08***	281.00	281.00	281.00	10/10
a2–24	0.15	**346.21***	2.68	**346.21***	**346.21***	**346.21***	346.59	347.97	10/10
a3–18	0.08	**236.82***	0.42	**236.82***	**236.82***	238.73	238.75	238.84	10/10
a3–24	0.23	**274.80***	0.97	**274.80***	**274.80***	275.18	275.18	275.18	10/10
a3–30	1.70	**413.27***	0.90	**413.27***	**413.27***	414.88	414.88	414.88	10/10
a3–36	1.78	**481.17***	2.54	**481.17***	**481.17***	483.86	484.05	485.71	10/10
a4–16	0.06	**222.49***	0.32	**222.49***	**222.49***	**222.49***	**222.49***	**222.49***	10/10
a4–24	0.52	**310.84***	0.49	**310.84***	**310.84***	311.48	311.48	311.48	10/10
a4–32	10.20	**393.96***	0.87	**393.96***	395.12	394.66	394.79	395.10	10/10
a4–40	8.62	**453.84***	1.53	**453.84***	459.42	456.93	457.08	457.84	10/10
a4–48	120	**554.54**	2.36	555.93	561.26	557.24	557.94	559.94	10/10
a5–40	19.03	**414.51***	1.08	414.80	420.35	415.62	415.65	415.86	10/10
a5–50	120	**559.17**	2.29	561.41	570.58	560.07	560.66	562.14	10/10
				r = 0.4					
a2–16	0.03	**237.38***	0.88	**237.38***	**237.38***	238.20	238.20	238.20	10/10
a2–20	0.83	**280.70***	2.35	**280.70***	**280.70***	282.90	283.24	283.38	10/10
a2–24	0.42	348.04 (347.04)**	3.85	**347.04***	**347.04***	**347.04***	349.67	350.84	10/10
a3–18	0.07	**236.82***	0.44	**236.82***	**236.82***	238.73	238.73	238.83	10/10
a3–24	0.28	**274.80***	1.13	**274.80***	**274.80***	275.58	275.58	275.58	10/10
a3–30	1.65	413.37 (413.34)**	1.48	**413.34***	**413.34***	415.51	415.75	416.06	10/10
a3–36	5.11	484.14**	2.63	**483.06**	483.86	485.98	487.42	489.81	10/10
a4–16	0.09	**222.49***	0.32	**222.49***	**222.49***	**222.49***	**222.49***	**222.49***	10/10
a4–24	0.66	**311.03***	0.53	**311.03***	311.65	311.48	311.48	311.48	10/10
a4–32	11.36	**394.26***	1.05	**394.26***	397.21	394.96	395.45	395.67	10/10
a4–40	6.96	**453.84***	1.94	**453.84***	459.46	457.01	457.39	457.92	10/10
a4–48	120	**554.60**	2.96	558.11	563.47	557.56	560.09	563.61	10/10
a5–40	20.35	**414.51***	1.21	416.25	420.32	415.63	415.63	415.63	10/10
a5–50	120	560.50	2.71	567.54	574.56	**560.41**	562.55	565.30	10/10

Table 6
Results on Cordeau instances with limited CS visits and $r = 0.7$.

Instance	e-ADARP2 (Bongiovanni et al., 2019)		DA (Su et al., 2023)			BI-LNS			Feasible
	RT [min]	Obj	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Obj max	
a2–16	0.09	**240.66***	1.60	**240.66***	**240.66***	242.83	245.50	247.39	10/10
a2–20	120	NA	2.88	**293.27**	294.11	NA	NA	NA	0/10
a2–24	16.02	358.21 (353.18)**	3.44	**353.18***	-	356.99	363.04	368.96	10/10
a3–18	0.80	**240.58***	0.97	**240.58***	**240.58***	242.49	246.13	250.30	10/10
a3–24	2.54	277.72 (275.97)**	2.06	**275.97***	277.43	277.52	277.52	277.52	10/10
a3–30	120	NA	1.30	**424.93**	436.20	432.27	436.56	441.08	10/10
a3–36	120	**494.04**	2.09	**494.04**	502.27	496.75	500.84	502.32	10/10
a4–16	1.12	**223.13***	0.52	**223.13***	**223.13***	**223.13***	223.95	227.99	10/10
a4–24	30.58	318.21 (316.65)**	0.90	**316.65***	318.31	319.37	321.10	324.19	10/10
a4–32	120	430.07	1.19	**397.87**	405.85	401.97	402.59	402.66	10/10
a4–40	120	NA	1.91	479.02	-	**471.72**	478.93	490.37	10/10
a4–48	120	NA	2.74	582.22	-	**579.71**	588.48	602.09	10/10
a5–40	120	447.63	1.63	424.26	436.94	**420.20**	423.88	428.17	10/10
a5–50	120	NA	2.64	603.24	-	**593.71**	602.30	612.16	9/10

Table 7
Results on Cordeau instances with unlimited CS visits and $r = 0.7$.

Instance	DA (Su et al., 2023)			BI-LNS			
	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Obj max	Feasible
a2–16	1.99	**240.66**	240.66	242.44	242.44	242.44	10/10
a2–20	5.27	**286.32**	288.89	290.33	291.23	293.18	10/10
a2–24	5.96	**354.38**	374.68	354.53	356.89	358.97	10/10
a3–18	1.10	**238.82**	238.82	241.95	242.46	243.04	10/10
a3–24	2.50	**275.20**	275.20	277.52	278.02	278.96	10/10
a3–30	2.85	**415.71**	417.07	419.16	426.30	433.20	10/10
a3–36	5.72	**484.85**	487.91	490.26	492.79	496.45	10/10
a4–16	0.52	**222.49**	**222.49**	**222.49**	223.57	225.06	10/10
a4–24	1.18	**315.98**	317.99	316.51	318.38	319.29	10/10
a4–32	2.06	**394.94**	401.82	396.64	397.98	400.26	10/10
a4–40	3.77	**458.52**	467.60	461.16	461.91	462.72	10/10
a4–48	6.72	568.08	575.96	**568.01**	570.80	575.03	10/10
a5–40	2.50	419.33	425.29	**418.79**	421.06	423.08	10/10
a5–50	5.88	579.15	588.98	**571.37**	575.49	579.58	10/10

Table 8
Statistics to BI-LNS optimizations on Cordeau instances with $r = 0.4$ and $r = 0.7$: Total number I^{total} of outer-level iterations, first outer-level iteration I^{feas} yielding a feasible solution, last outer-level iteration I^{last} yielding an improvement, and time T^{last} of last improvement. The values are averaged over all trials, which yielded a feasible solution.

Instance	Limited CS visits				Unlimited CS visits			
	I^{total}	I^{feas}	I^{last}	T^{last} [min]	I^{total}	I^{feas}	I^{last}	T^{last} [min]
				r=0.4				
a2–16	13730	5	925	0.35	13440	7	1244	0.44
a2–20	7679	4	2958	1.87	8423	3	4706	2.85
a2–24	12648	34	5302	2.04	10013	71	3372	1.68
a3–18	5248	4	449	0.40	5479	4	199	0.18
a3–24	2941	3	906	1.53	2930	4	482	0.76
a3–30	9177	7	4395	2.49	9025	10	943	0.47
a3–36	6491	30	2685	2.12	4500	45	2429	2.52
a4–16	6681	3	429	0.32	6483	7	1942	1.49
a4–24	3218	1	801	1.24	3230	4	879	1.35
a4–32	2288	4	837	1.83	2304	3	1346	2.88
a4–40	1862	2	760	2.03	1791	3	694	1.92
a4–48	1858	3	1316	3.24	1960	2	938	2.45
a5–40	1256	1	136	0.54	1258	2	114	0.44
a5–50	1348	1	377	1.34	1303	2	642	2.40
				r=0.7				
a2–16	39385	326	4818	0.57	37193	167	6637	0.92
a2–20	NA	NA	NA	NA	20224	1806	11358	2.50
a2–24	43320	2668	7566	0.87	16292	461	11158	3.12
a3–18	22488	77	4627	0.97	16328	76	7041	1.91
a3–24	8656	14	2151	1.07	9365	30	4075	1.93
a3–30	37594	2047	13183	1.99	12178	478	6206	2.44
a3–36	16364	153	5427	1.56	8935	74	3731	2.07
a4–16	13596	7	5523	2.08	11988	10	5720	2.38
a4–24	13133	46	5328	2.18	11541	56	4965	2.37
a4–32	18876	323	2430	0.60	7496	65	5734	3.75
a4–40	15751	194	2145	0.63	3770	76	2573	3.28
a4–48	17202	1546	10220	2.75	3197	60	2526	3.80
a5–40	7279	28	3690	2.70	2866	23	1525	2.62
a5–50	24827	3359	11348	2.17	3047	130	2233	3.45

$r = 0.4$, the results of DA and BI-LNS on the Cordeau instances are very similar to the results with limited CS visits. The same holds for BI-LNS for $r = 0.1$ (there are no results of DA on Cordeau instances with unlimited CS visits with $r = 0.1$ in (Su et al., 2023)). The detailed results with $r = 0.4$ and $r = 0.1$ can be found in Tables D.18 and D.19 in the appendix. The results of DA and BI-LNS on the Cordeau instances with $r = 0.7$ and unlimited CS visits are shown in Table 7. On most problem instances the results of both approaches are better compared to the results with limited CS visits in Table 6. Again, on the smaller instances, DA outperforms BI-LNS, but on the five largest problem instances, BI-LNS yields better average results.

In order to gain more insights into the optimizations with BI-LNS, we recorded different statistics to the optimization runs: The total number I^{total} of executed iterations at the outer level, the outer-level iteration I^{feas} in which the first feasible solution was found, the outer-level iteration I^{last} in which the last improvement was found, and the runtime T^{last} until the last improvement was found. The statistics on the Cordeau instances with $r = 0.1$ are shown in Table D.20 in the appendix and for $r = 0.4$ and $r = 0.7$ in Table 8. The statistics are averaged over all trials, which yielded a feasible solution. With limited CS visits and $r = 0.4$, between 1000 and 14,000 outer-level iterations are executed within the time limit of five minutes. Not surprisingly, the number of executed iterations tends to decrease with an increasing problem size. Only a small number of outer-level iterations are necessary to find a feasible solution and it typically takes notably less than 5 min to find the final solution. With unlimited CS visits and $r = 0.4$ the statistics are very similar to those with limited CS visits and $r = 0.4$. With $r = 0.7$ and limited CS visits, it is harder to find a feasible solution. This does not only result in higher values for I^{feas} but also in higher values for I^{total} since at the inner level only a small number of requests are reinserted as long as no feasible solution is found. With unlimited CS visits and $r = 0.7$ the first feasible solution is typically found early and the last improvement later compared to limited CS visits and $r = 0.7$.

5.4.2. Results on Ropke instances

The results with DA and BI-LNS on the Ropke instances with limited charging station visits and $r = 0.4$ are shown in Table 9. It can be seen that BI-LNS clearly outperforms DA on these instances. In all trials with BI-LNS, a feasible solution is found, while DA is not able to find a feasible solution for the a8-96 instance in 50 trials. On six of the 10 problem instances, the average result of BI-LNS is better than the best result of DA. Furthermore, compared to the Cordeau instances, the runtime of DA is notably higher and often more than the five minute runtime of BI-LNS. The results with limited CS visits and $r = 0.7$ are shown in Table 10. Su et al. (2023) state that DA was not able to solve any of the Ropke instances with limited CS visits and a high energy restriction of $r = 0.7$. BI-LNS is also not able to solve the instances with 12 requests per vehicle. However, for four of the six other instances it yielded a feasible solution in all trials and on the a6-60 instance nine of the 10 trials yielded a feasible solution.

The results on the Ropke instances with unlimited CS visits and $r = 0.4$ and $r = 0.7$, are shown in Tables 11 and 12, respectively. Both DA and BI-LNS find feasible solutions in all trials on all problem instances. For most problem instances, BI-LNS finds better solutions than DA and on all problem instances, the worst result of BI-LNS is better than the average result of DA.

Thus, while DA outperforms BI-LNS on most of the Cordeau instances, it cannot compete with BI-LNS on the larger Ropke instances. This demonstrates the good scalability of BI-LNS.

Analogous to the Cordeau instances, we collected statistics to the BI-LNS optimizations on the Ropke instances. The statistics can be seen in Table 13. A comparison between the statistics with $r = 0.4$ and $r = 0.7$ and with and without limitation of CS visits shows the same trend as on the Cordeau instances. The number of executed outer-level iterations is typically lower and the time of the last improvement is typically higher compared to the Cordeau instances. Especially with $r = 0.7$ and unlimited CS visits, the average time of the last improvement is near

Table 9
Results on Ropke instances with limited CS visits and $r = 0.4$.

Instance	DA (Su et al., 2023)			BI-LNS			
	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Obj max	Feasible
a5–60	4.89	697.97	718.44	**688.16**	693.43	696.81	10/10
a6–48	4.29	506.91	514.46	**506.85**	507.43	509.31	10/10
a6–60	2.89	694.78	706.07	**692.69**	695.13	699.78	10/10
a6–72	5.83	799.60	821.17	**771.97**	778.30	784.30	10/10
a7–56	1.67	**613.66**	624.40	614.58	616.02	618.34	10/10
a7–70	4.56	766.05	784.54	**761.62**	765.71	769.13	10/10
a7–84	9.74	932.12	-	**886.19**	891.04	896.35	10/10
a8–64	10.69	638.36	652.30	**637.95**	640.41	643.17	10/10
a8–80	7.47	811.19	833.05	**793.17**	798.41	804.88	10/10
a8–96	10.29	NA	-	**1048.72**	1057.47	1068.40	10/10

Table 10
Results on Ropke instances with limited CS visits and $r = 0.7$.

Instance	DA (Su et al., 2023)	BI-LNS			
	Obj min	Obj min	Obj mean	Obj max	Feasible
a5–60	NA	NA	NA	NA	0/10
a6–48	NA	**519.55**	522.50	528.81	10/10
a6–60	NA	**733.45**	742.02	754.03	9/10
a6–72	NA	NA	NA	NA	0/10
a7–56	NA	**649.11**	669.71	687.20	10/10
a7–70	NA	NA	NA	NA	0/10
a7–84	NA	NA	NA	NA	0/10
a8–64	NA	**646.82**	652.38	662.67	10/10
a8–80	NA	**854.85**	863.74	876.91	10/10
a8–96	NA	NA	NA	NA	0/10

Table 11
Results on Ropke instances with unlimited CS visits and $r = 0.4$.

Instance	DA (Su et al., 2023)			BI-LNS			
	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Obj max	Feasible
a5–60	4.75	691.72	709.78	**685.68**	689.42	693.28	10/10
a6–48	4.26	507.25	514.64	**506.85**	507.30	508.29	10/10
a6–60	2.90	692.83	701.86	**692.25**	693.68	696.44	10/10
a6–72	5.71	781.22	801.86	**774.38**	778.84	783.39	10/10
a7–56	1.65	615.74	623.51	**614.58**	615.40	616.31	10/10
a7–70	4.56	**761.58**	778.04	762.78	764.51	767.59	10/10
a7–84	7.61	896.91	916.23	**884.94**	890.81	896.54	10/10
a8–64	11.99	**637.84**	652.17	637.95	640.30	643.87	10/10
a8–80	7.52	813.16	829.92	**794.38**	797.08	800.00	10/10
a8–96	9.41	1058.41	1090.04	**1048.45**	1053.35	1059.35	10/10

Table 12
Results on Ropke instances with unlimited CS visits and $r = 0.7$.

Instance	DA (Su et al., 2023)			BI-LNS			
	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Obj max	Feasible
a5–60	8.21	708.54	723.73	**697.87**	709.11	719.19	10/10
a6–48	8.07	**509.76**	525.10	511.04	514.53	517.13	10/10
a6–60	4.83	**697.57**	711.52	699.70	705.56	714.29	10/10
a6–72	9.57	796.19	826.48	**788.34**	801.80	812.75	10/10
a7–56	3.53	**625.91**	641.82	627.34	633.38	636.99	10/10
a7–70	8.00	781.56	800.35	**777.69**	785.49	793.49	10/10
a7–84	11.75	915.61	938.49	**900.98**	916.93	925.47	10/10
a8–64	21.50	649.93	668.48	**645.62**	648.60	654.84	10/10
a8–80	12.41	843.26	865.90	**815.06**	825.74	838.01	10/10
a8–96	13.45	1097.76	1136.43	**1072.77**	1091.06	1117.57	10/10

to the time limit of 5 min for multiple problem instances. Hence, one can expect that with this setting BI-LNS could profit from a higher time limit.

The best result of BI-LNS for the a8-80 instance with limited CS visits and $r = 0.7$ is illustrated in Fig. 3. Shown are for each vehicle, the number of served requests, the driving time, the amount of charged energy and the driving/charging schedule. In this result, the excess ride time is 304.67 min and the total driving time of vehicles is 1038.25 min, yielding an objective value of 854.85. Since each charging station can be visited only once, only three vehicles charge. It can be seen that these vehicles have two to three times higher driving times than the vehicles that do not charge and that they serve more than half of the requests. The vehicles that do not charge are idle for most of the planning horizon. For comparison, Fig. 4 illustrates the best result with unlimited CS visits. In this result, the excess ride time reduces to 279.75 min and the total vehicles' driving time reduces to 993.50 min, resulting in an objective value of 815.06. The requests are more equally distributed over the vehicles compared to the result with limited CS visits. Vehicle 1 has only a short charging session at the end depot after

Table 13
Statistics to BI-LNS optimizations on Ropke instances: Total number I^{total} of outer-level iterations, first outer-level iteration I^{feas} yielding a feasible solution, last outer-level iteration I^{last} yielding an improvement, and time T^{last} of last improvement. The values are averaged over all trials, which yielded a feasible solution.

Instance	Limited CS visits				Unlimited CS visits			
	I^{total}	I^{feas}	I^{last}	T^{last} [min]	I^{total}	I^{feas}	I^{last}	T^{last} [min]
				r=0.4				
a5–60	2492	7	1084	2.37	2380	7	1257	3.11
a6–48	972	1	333	1.69	990	4	341	1.65
a6–60	1216	2	295	1.18	1234	3	610	2.60
a6–72	1603	6	893	2.67	1547	2	873	2.74
a7–56	876	1	420	2.37	894	2	470	2.62
a7–70	789	1	287	1.81	786	2	459	2.89
a7–84	1333	5	732	2.80	1166	1	568	2.49
a8–64	671	1	289	2.12	673	2	333	2.42
a8–80	520	1	178	1.72	507	1	283	2.78
a8–96	1406	7	767	2.88	1048	1	853	3.97
				r=0.7				
a5–60	NA	NA	NA	NA	3156	162	2710	4.15
a6–48	10508	115	3028	1.41	2860	42	2133	3.26
a6–60	22883	8034	17043	3.34	2646	45	2266	4.10
a6–72	NA	NA	NA	NA	2406	91	1744	3.52
a7–56	20946	4518	12795	2.86	2554	24	1607	2.77
a7–70	NA	NA	NA	NA	1928	47	1579	4.17
a7–84	NA	NA	NA	NA	1923	83	1581	3.96
a8–64	6438	118	3658	2.72	2094	8	1331	2.86
a8–80	11512	3887	7416	2.89	1332	17	1089	3.79
a8–96	NA	NA	NA	NA	2223	200	1974	4.39

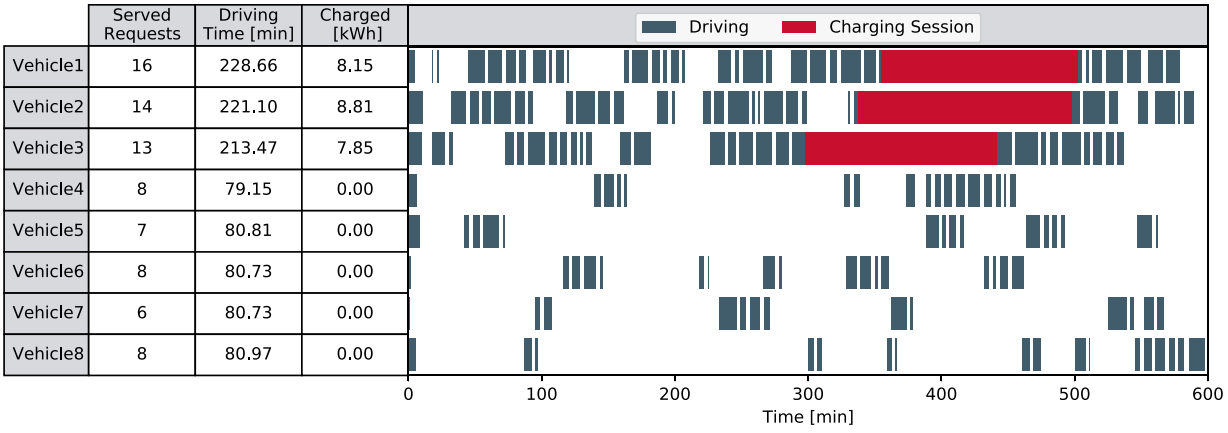

Fig. 3. Detailed result on a8-80 instance with $r = 0.7$ and limited CS visits.

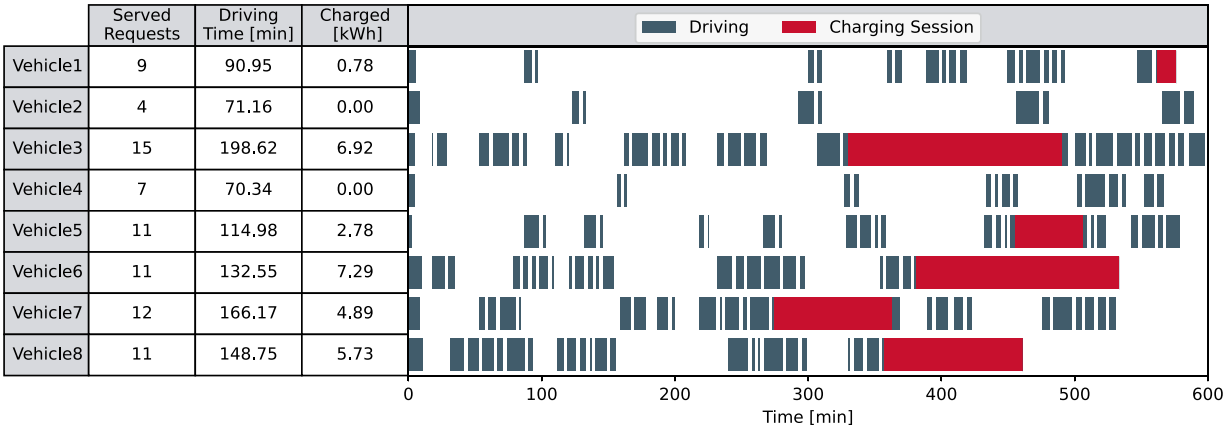

Fig. 4. Detailed result on a8-80 instance with $r = 0.7$ and unlimited CS visits.

Table 14
Results of BI-LNS on large instances with unlimited CS visits and $r = 0.7$.

Time limit [min]	Obj min	Obj mean	Obj max	Feasible	Gap Mean
		a180-3600			
5	29,793.51	29,977.80	30,145.25	10/10	–
10	29,397.03	29,546.14	29,768.16	10/10	−1.44%
15	29,177.96	29,312.55	29,587.32	10/10	−2.22%
		a200-4000			
5	32,803.99	33,161.17	33,485.56	10/10	–
10	32,390.78	32,558.80	32,751.15	10/10	−1.82%
15	32,013.56	32,263.01	32,530.51	10/10	−2.71%
		a220-4400			
5	36,365.84	37,523.84	39,556.70	10/10	–
10	35,627.80	35,746.51	35,917.44	10/10	−4.74%
15	35,259.06	35,428.04	35,567.21	10/10	−5.59%
		a240-4800			
5	41,357.57	42,150.46	42,943.32	2/10	–
10	38,496.01	38,817.89	39,084.53	10/10	−7.91%
15	38,270.98	38,460.95	38,579.35	10/10	−8.75%
		a260-5200			
5	NA	NA	NA	0/10	–
10	41,890.96	42,191.50	42,490.99	10/10	NA
15	41,472.11	41,745.98	41,855.53	10/10	NA

serving all its requests in order to satisfy the minimum final battery capacity ratio.

5.4.3. Results on large instances

In order to investigate the limits of the scalability of BI-LNS, experiments on five large problem instances with 180 to 260 vehicles and 3600 to 5200 requests were executed. The experiments were run not only with a time limit of 5 min per trial but also with time limits of 10 min and 15 min to investigate the convergence of the approach. The results are shown in Table 14 For time limits of 10 and 15 min, the percentage gaps between the corresponding mean results and the mean results with five minute time limit are shown. For the instances with 3600, 4000, and 4400 requests, BI-LNS finds a feasible solution within five minutes in all trials. With time limits of 10 and 15 min, better results are achieved and the gaps increase with an increasing problem size. On the problem instance with 4800 requests, only two trials with a time limit of five minutes yielded a feasible solution and the gaps to the results with 10 and 15 min time limit are large. For the a260-5200 instance, BI-LNS was no longer able to find a feasible solution in any trial. Thus, one can conclude that with around 4000 requests, the BI-LNS approach reaches its limits — at least with a time limit of five minutes. However, being able to schedule around 4000 requests within five minutes, can be assumed to be a reasonable performance for many practical applications and, as one can see, with a higher time budget, even more requests can be scheduled.

Table 15 shows statistics to the BI-LNS optimizations on the large instances. The number of executed outer-level iterations is in a range between 270 and 410. After a first feasible solution is found, only a small number of further iterations is executed, even with a time limit of 15 min. The reason is that at the inner level the number of reinsertions increases after a feasible solution is found and this makes the inner level very time consuming on the large instances. According to the results of the parameter tuning (see Section 5.2), the number Q of reinsertions after a feasible solution is found has been set to 19. With a lower value, more outer-level iterations could be executed, which might improve the results on the large instances. In order to investigate whether this is the case, the optimizations on the a200-4000 instance with a time limit of 5 min were executed with different values for Q. The results are shown in Table 16. One can see, that a lower value for Q is not beneficial. On the contrary, higher values can slightly improve the results compared to $Q = 19$.

6. Summary and conclusion

The present work proposes a bilevel large neighborhood search heuristic (BI-LNS) for the challenging electric autonomous dial-a-ride problem (E-ADARP). The approach does not rely on a compute-intensive operator for the insertion of charging sessions since it inserts charging sessions in empty routes before it inserts the pick-up and drop-off locations. Numerical experiments have shown that in this way a considerable scalability is achieved. On benchmark instances with up to five vehicles and 50 requests the proposed approach was compared to the exact 2-index mixed integer linear programming approach (e-ADARP2) of Bongiovanni et al. (2019) and the deterministic annealing heuristic (DA) of Su et al. (2023). On a number of instances, the proposed approach was able to yield better results than the other two approaches while being competitive on the rest of the instances. In further experiments on a set of larger problem instances with up to 8 vehicles and 96 requests, BI-LNS significantly outperformed DA and was able to find feasible solutions in most of the trials. On a set of large benchmark instances with up to 260 vehicles and 5200 requests, the scalability of the BI-LNS approach was investigated and it could be shown that the approach scales up to around 4000 requests, which can be assumed to be sufficient for a wide range of practical applications.

Different potential improvements of the proposed approach could be investigated as part of future research. One of such improvements could be the use of adaptive large neighborhood search, which selects in each iteration an operator from different alternatives and dynamically adapts the probabilities of the individual operators for being selected. At the outer level, BI-LNS already selects between three variation operators with different (so far fixed) probabilities. Additional operators could be considered, for example, an operator that removes a charging session from one route and inserts a new charging session in another route or an operator that modifies more than one charging session. A further potential improvement could be the combination of BI-LNS with linear programming to a hybrid approach. At the outer level, the service times of new solutions could be improved with help of linear programming in order to reduce the excess ride times of passengers.

Table 15
Statistics to BI-LNS optimizations on large instances: Total number I^{total} of outer-level iterations, first outer-level iteration I^{feas} yielding a feasible solution, last outer-level iteration I^{last} yielding an improvement, and time T^{last} of last improvement. The values are averaged over all trials, which yielded a feasible solution.

Time limit [min]	I^{total}	I^{feas}	I^{last}	T^{last} [min]
a180-3600				
5	270	264	269	4.72
10	282	264	281	9.52
15	296	264	295	14.63
a200-4000				
5	289	286	289	4.75
10	300	286	299	9.59
15	311	286	311	14.53
a220-4400				
5	311	310	311	4.81
10	321	310	321	9.50
15	332	310	331	14.56
a240-4800				
5	334	334	334	4.98
10	376	369	376	9.68
15	386	369	385	14.53
a260-5200				
5	NA	NA	NA	NA
10	399	393	398	9.45
15	407	393	407	14.55

Table 16
Mean objective over ten trials on the a200-4000 with a time limit of 5 min resulting from different settings of Q with $I = 800$ and $p_{mod} = 0.55$.

Q	3	11	19	27	35	43
Mean Obj	33,518.86	33,314.86	33,161.17	33,095.75	33,012.34	33,043.75

Table C.17
Characteristics of the tuning problem instances.

Instance	K	N	r	T [min]
1	8	96	0.60	720
2	3	36	0.46	720
3	5	50	0.48	720
4	6	60	0.74	720
5	5	60	0.70	720
6	3	36	0.64	720
7	2	16	0.48	720
8	3	36	0.49	720
9	3	24	0.63	720
10	7	70	0.57	720

Table D.18
Results of BI-LNS on Cordeau instances with unlimited CS visits and $r = 0.1$.

Instance	Obj min	Obj mean	Obj max	Feasible
a2–16	238.20	238.20	238.20	10/10
a2–20	281.00	281.00	281.00	10/10
a2–24	346.21	346.22	346.28	10/10
a3–18	238.73	238.73	238.73	10/10
a3–24	275.18	275.18	275.18	10/10
a3–30	414.88	414.88	414.88	10/10
a3–36	483.86	483.86	483.86	10/10
a4–16	222.49	222.49	222.49	10/10
a4–24	311.48	311.48	311.48	10/10
a4–32	394.66	394.79	395.10	10/10
a4–40	456.93	457.37	457.84	10/10
a4–48	557.25	557.82	559.26	10/10
a5–40	415.62	415.62	415.62	10/10
a5–50	560.07	560.95	562.15	10/10

Another practical extension of BI-LNS would be an approach for the automated setting of the initial number C^{init} of charging sessions per vehicle based on characteristics of the problem instance.

CRediT authorship contribution statement

Steffen Limmer: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Roles/Writing – original draft, Writing – review & editing.

Declaration of competing interest

The author has no conflict of interest to declare.

Data availability

Data will be made available on request.

Appendix A. Insertion operator

Algorithm 4 outlines the used insertion operator.

Appendix B. MILP approach

For the computation of the corrected global optima, we used the following mixed integer linear programming formulation of the problem:

$$\min w_1 \sum_{i,j \in \mathcal{V}} \sum_{k \in \mathcal{K}} t_{ij} x_{i,j}^k + w_2 \sum_{i \in \mathcal{P}^+} R_i \tag{B.1}$$

Algorithm 4: Insertion Operator.

Input: routes *rt*, *new_requests*
Output: routes rt^*, rejected requests *rej_req*

$rt^* = rt$;
$rej\_req = \emptyset$;
for *req* in *new_requests* **do**
  $best\_vehicle = 0$;
  $best\_route = []$;
  $best\_obj\_inc = \infty$;
  for V in vehicles **do**
    $route = rt^*[V]$;
    for i in $[2, \dots, route.size - 1]$ **do**
      $route2$ = insert(*route*,*req.pickup*,*i*);
      if feasible*(route2)* **then**
        for j in $[i+1, \dots, route2.size - 1]$ **do**
          $route3$ = insert(*route2*,*req.dropoff*,*j*);
          if feasible*(route3)* **then**
            obj_inc = objective(*route3*)-objective(*route*)+random_noise();
            if $obj\_inc < best\_obj\_inc$ **then**
              $best\_obj\_inc = obj\_inc$;
              $best\_vehicle = V$;
              $best\_route = route3$;
            end
          end
        end
      end
    end
  end
  if $best\_obj\_inc \neq \infty$ **then**
    $rt^*[best\_vehicle] = best\_route$;
  else
    $rej\_req = rej\_req \cup \{req\}$;
  end
end
return rt^*, *rej_req*;

subject to:

$$\sum_{j \in \mathcal{P}^+ \cup C \cup \{2N+K+k\}} x^k_{2N+k,j} = 1 \qquad \forall k \in \mathcal{K}, \tag{B.2}$$

$$\sum_{i \in \mathcal{P}^- \cup C \cup \{2N+k\}} x^k_{i,2N+K+k} = 1 \qquad \forall k \in \mathcal{K}, \tag{B.3}$$

$$x^k_{i,j} = 0 \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{V}, \forall j \in \mathcal{D}^+, \tag{B.4}$$

$$x^k_{i,j} = 0 \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{V}, \forall j \in \mathcal{D}^- \setminus \{2N+K+k\}, \tag{B.5}$$

$$x^k_{i,j} = 0 \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{D}-, \forall j \in \mathcal{V}, \tag{B.6}$$

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}} x^k_{i,j} \leq 1 \qquad \forall j \in \mathcal{V}, \tag{B.7}$$

$$\sum_{j \in \mathcal{V}} x^k_{j,i} - \sum_{j \in \mathcal{V}} x^k_{i,j} = 0 \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{V} \setminus \mathcal{D}^-, \tag{B.8}$$

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}} x^k_{i,j} = 1 \qquad \forall j \in \mathcal{P}^+, \tag{B.9}$$

$$\sum_{i \in \mathcal{V}} x^k_{i,j} = \sum_{i \in \mathcal{V}} x^k_{i,j+N} \qquad \forall k \in \mathcal{K}, \forall j \in \mathcal{P}^+, \tag{B.10}$$

$$w^-_i \leq S_i \leq w^+_i \qquad \forall i \in \mathcal{V}, \tag{B.11}$$

$$S_i \geq S_j + d_j + t_{j,i} - M_{j,i} \cdot (1 - \sum_{k \in \mathcal{K}} x^k_{j,i}) \qquad \forall i, j \in \mathcal{V}, \tag{B.12}$$

$$S_i \leq S_{i+N} \qquad \forall i \in \mathcal{P}^+, \tag{B.13}$$

$$S_{i+N} - (S_i + d_i) \leq S \qquad \forall i \in \mathcal{P}^+, \tag{B.14}$$

$$L_i \geq L_j + l_i - O_{j,i} \cdot (1 - \sum_{k \in \mathcal{K}} x^k_{j,i}) \qquad \forall i, j \in \mathcal{V}, \tag{B.15}$$

$$L_i = 0 \qquad \forall i \in \mathcal{D}^+ \cup C, \tag{B.16}$$

$$L_i \leq L \qquad \forall i \in \mathcal{V}, \tag{B.17}$$

$$B^{arr}_i = B \qquad \forall i \in \mathcal{D}^+, \tag{B.18}$$

$$B^{arr}_i \leq B^{arr}_j - e_{j,i} + B \cdot (1 - \sum_{k \in \mathcal{K}} x^k_{j,i}) \qquad \forall i \in \mathcal{V}, \forall j \in \mathcal{V} \setminus C, \tag{B.19}$$

$$B^{arr}_i \geq B^{arr}_j - e_{j,i} - B \cdot (1 - \sum_{k \in \mathcal{K}} x^k_{j,i}) \qquad \forall i \in \mathcal{V}, \forall j \in \mathcal{V} \setminus C, \tag{B.20}$$

$$B^{arr}_i \leq B^{arr}_j - E_j \cdot \alpha_j - e_{j,i} + B \cdot (1 - \sum_{k \in \mathcal{K}} x^k_{j,i}) \qquad \forall i \in \mathcal{V} \setminus \mathcal{P}^-, \forall jC, \tag{B.21}$$

$$B^{arr}_i \geq B^{arr}_j - E_j \cdot \alpha_j - e_{j,i} - B \cdot (1 - \sum_{k \in \mathcal{K}} x^k_{j,i}) \qquad \forall i \in \mathcal{V} \setminus \mathcal{P}^-, \forall jC, \tag{B.22}$$

$$B^{arr}_i + E_i \cdot \alpha_i \leq B \qquad \forall i \in C, \tag{B.23}$$

$$0 \leq B^{arr}_i \leq B \qquad \forall i \in \mathcal{V}, \tag{B.24}$$

$$B^{arr}_i \geq r \cdot B \qquad \forall i \in \mathcal{D}^-, \tag{B.25}$$

$$R_i = S_{i+N} - (S_i + d_i) - t_{i,i+N} \qquad \forall i \in \mathcal{P}^+, \tag{B.26}$$

where $M_{j,i}$ in Constraint (B.12) is set to $\max\{0, w^+_j + d_j + t_{j,i} - w^-_i\}$ and $O_{j,i}$ in Constraint (B.15) is set to $\min\{L, L+l_j\}$. In order to accelerate the solving process, we apply tighter bounds for service times and vehicle loads, and arc elimination (Cordeau, 2006). Furthermore, we use results of BI-LNS as initial solutions. We used version 9.1.0 of Gurobi[3] as solver.

Appendix C. Tuning instances

The characteristics of the 10 problem instances used for tuning of the algorithm parameters are shown in Table C.17.

Appendix D. Additional results on Cordeau instances

Table D.18 shows the results of BI-LNS on the Cordeau instances with unlimited CS visits for $r = 0.1$ and Table D.19 shows the results of DA and BI-LNS on the Cordeau instances with unlimited CS visits for $r = 0.4$. Table D.20 shows statistics to the BI-LNS optimizations on the Cordeau instances with $r = 0.1$.

[3] https://www.gurobi.com/

Table D.19
Results of DA and BI-LNS on Cordeau instances with unlimited CS visits $r = 0.4$.

Instance	DA (Su et al., 2023)			BI-LNS			
	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Obj max	Feasible
a2–16	0.84	**237.38**	**237.38**	238.20	238.20	238.20	10/10
a2–20	2.42	**280.70**	**280.70**	282.62	283.10	283.38	10/10
a2–24	4.43	346.28	**346.28**	347.93	349.07	349.82	10/10
a3–18	0.42	**236.82**	**236.82**	238.73	238.73	238.73	10/10
a3–24	1.11	**274.80**	**274.80**	275.18	275.46	275.58	10/10
a3–30	1.73	**413.34**	**413.34**	415.51	415.81	415.85	10/10
a3–36	4.15	**481.17**	**481.17**	484.07	485.37	487.74	10/10
a4–16	0.30	**222.49**	**222.49**	**222.49**	**222.49**	**222.49**	10/10
a4–24	0.49	**311.03**	311.65	311.48	311.48	311.48	10/10
a4–32	1.03	**394.26**	397.27	394.96	395.15	395.41	10/10
a4–40	2.02	**453.84**	458.74	456.93	457.50	457.92	10/10
a4–48	3.86	558.96	564.86	**557.63**	559.63	562.58	10/10
a5–40	1.18	415.79	419.82	**415.62**	**415.62**	**415.62**	10/10
a5–50	3.07	567.13	574.28	**560.41**	562.64	567.91	10/10

Table D.20
Statistics to BI-LNS optimizations on Cordeau instances with $r = 0.1$: Total number I^{total} of outer-level iterations, first outer-level iteration I^{feas} yielding a feasible solution, last outer-level iteration I^{last} yielding an improvement, and time T^{last} of last improvement. The values are averaged over all trials, which yielded a feasible solution.

Instance	Limited CS visits				Unlimited CS visits			
	I^{total}	I^{feas}	I^{last}	T^{last} [min]	I^{total}	I^{feas}	I^{last}	T^{last} [min]
a2–16	10468	4	12	0.00	10718	7	10	0.01
a2–20	3292	2	14	0.02	3330	3	10	0.01
a2–24	4334	3	658	0.72	4353	5	716	0.80
a3–18	4721	4	686	0.72	4858	4	42	0.04
a3–24	2454	3	367	0.73	2510	3	641	1.27
a3–30	2732	4	188	0.34	2885	4	424	0.71
a3–36	2352	5	221	0.45	2408	5	278	0.54
a4–16	6480	3	429	0.32	6693	7	1942	1.44
a4–24	2819	1	662	1.16	2905	4	998	1.70
a4–32	1910	2	449	1.18	1899	3	540	1.42
a4–40	1136	1	476	2.09	1137	2	442	1.92
a4–48	871	1	390	2.23	877	2	332	1.86
a5–40	1115	1	225	1.02	1119	2	110	0.48
a5–50	879	1	307	1.72	862	2	136	0.76

References

Bongiovanni, C., 2020. The Electric Autonomous Dial-A-Ride Problem (Ph.D. thesis). EPFL, Lausanne, http://dx.doi.org/10.5075/epfl-thesis-7431.

Bongiovanni, C., Kaspi, M., Geroliminis, N., 2019. The electric autonomous dial-a-ride problem. Transp. Res. B 122, 436–456. http://dx.doi.org/10.1016/j.trb.2019.03.004.

Braekers, K., Caris, A., Janssens, G.K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transp. Res. B 67, 166–186. http://dx.doi.org/10.1016/j.trb.2014.05.007.

Cordeau, J.-F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Oper. Res. 54 (3), 573–586. http://dx.doi.org/10.1287/opre.1060.0283.

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. B 37 (6), 579–594. http://dx.doi.org/10.1016/S0191-2615(02)00045-0.

Garaix, T., Artigues, C., Feillet, D., Josselin, D., 2011. Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. Comput. Oper. Res. 38 (10), 1435–1442. http://dx.doi.org/10.1016/j.cor.2010.12.014.

Gschwind, T., Drexl, M., 2019. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. Transp. Sci. 53 (2), 480–491. http://dx.doi.org/10.1287/trsc.2018.0837.

Gschwind, T., Irnich, S., 2015. Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. Transp. Sci. 49 (2), 335–354. http://dx.doi.org/10.1287/trsc.2014.0531.

Ho, S.C., Szeto, W., Kuo, Y.-H., Leung, J.M., Petering, M., Tou, T.W., 2018. A survey of dial-a-ride problems: Literature review and recent developments. Transp. Res. B 111, 395–421. http://dx.doi.org/10.1016/j.trb.2018.02.001.

Hoché, T., Barth, D., Mautor, T., Burghout, W., 2020. Charging management of shared taxis: Neighbourhood search for the E-ADARP. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems. ITSC, pp. 1–6. http://dx.doi.org/10.1109/ITSC45102.2020.9294446.

Jorgensen, R.M., Larsen, J., Bergvinsdottir, K.B., 2007. Solving the dial-a-ride problem using genetic algorithms. J. Oper. Res. Soc. 58 (10), 1321–1331. http://dx.doi.org/10.1057/palgrave.jors.2602287.

Liang, X., de Almeida Correia, G.H., An, K., van Arem, B., 2020. Automated taxis' dial-a-ride problem with ride-sharing considering congestion-based dynamic travel times. Transp. Res. C 112, 260–281. http://dx.doi.org/10.1016/j.trc.2020.01.024.

Masmoudi, M.A., Braekers, K., Masmoudi, M., Dammak, A., 2017. A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. Comput. Oper. Res. 81, 1–13. http://dx.doi.org/10.1016/j.cor.2016.12.008.

Masmoudi, M.A., Hosny, M., Braekers, K., Dammak, A., 2016. Three effective meta-heuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. Transp. Res. E 96, 60–80. http://dx.doi.org/10.1016/j.tre.2016.10.002.

Masmoudi, M.A., Hosny, M., Demir, E., Genikomsakis, K.N., Cheikhrouhou, N., 2018. The dial-a-ride problem with electric vehicles and battery swapping stations. Transp. Res. E 118, 392–420. http://dx.doi.org/10.1016/j.tre.2018.08.005.

Molenbruch, Y., Braekers, K., Caris, A., 2017a. Benefits of horizontal cooperation in dial-a-ride services. Transp. Res. E 107, 97–119. http://dx.doi.org/10.1016/j.tre.2017.09.001.

Molenbruch, Y., Braekers, K., Caris, A., 2017b. Typology and literature review for dial-a-ride problems. Ann. Oper. Res. 259, 295–325. http://dx.doi.org/10.1007/s10479-017-2525-0.

Narayanan, S., Chaniotakis, E., Antoniou, C., 2020. Shared autonomous vehicle services: A comprehensive review. Transp. Res. C 111, 255–293. http://dx.doi.org/10.1016/j.trc.2019.12.008.

Parragh, S.N., 2011. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Transp. Res. C 19 (5), 912–930. http://dx.doi.org/10.1016/j.trc.2010.06.002.

Parragh, S.N., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 40 (1), 490–497. http://dx.doi.org/10.1016/j.cor.2012.08.004.

Pimenta, V., Quilliot, A., Toussaint, H., Vigo, D., 2017. Models and algorithms for reliability-oriented dial-a-ride with autonomous electric vehicles. European J. Oper. Res. 257 (2), 601–613. http://dx.doi.org/10.1016/j.ejor.2016.07.037.

Ropke, S., Cordeau, J.-F., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transp. Sci. 43 (3), 267–286. http://dx.doi.org/10.1287/trsc.1090.0272.

Ropke, S., Cordeau, J.-F., Laporte, G., 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks 49 (4), 258–272. http://dx.doi.org/10.1002/net.20177.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472. http://dx.doi.org/10.1287/trsc.1050.0135.

Shaheen, S., Cohen, A., 2020. Chapter 3 - Mobility on demand (MOD) and mobility as a service (MaaS): Early understanding of shared mobility impacts and public transit partnerships. In: Antoniou, C., Efthymiou, D., Chaniotakis, E. (Eds.), Demand for Emerging Transportation Systems. Elsevier, pp. 37–59. http://dx.doi.org/10.1016/B978-0-12-815018-4.00003-6.

Shaw, P., 1997. A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems. Technical Report, University of Strathclyde.

Su, Y., Dupin, N., Puchinger, J., 2023. A deterministic annealing local search for the electric autonomous dial-a-ride problem. European J. Oper. Res. http://dx.doi.org/10.1016/j.ejor.2023.02.012.

Vallée, S., Oulamara, A., Chérif-Khettaf, W.R., 2017. Maximizing the number of served requests in an online shared transport system by solving a dynamic DARP. In: Computational Logistics - 8th International Conference, ICCL 2017, Southampton, UK, October 18–20, 2017, Proceedings. In: Lecture Notes in Computer Science, vol. 10572, Springer, pp. 64–78. http://dx.doi.org/10.1007/978-3-319-68496-3_5.

Venkatraman, P., Levin, M.W., 2019. A congestion-aware tabu search heuristic to solve the shared autonomous vehicle routing problem. J. Intell. Transp. Syst. 25 (4), 343–355. http://dx.doi.org/10.1080/15472450.2019.1665521.