Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Letting a Large Neighborhood Search for an Electric Dial-A-Ride Problem Fly: On-The-Fly Charging Station Insertion

Maria Bresich, Guenther Raidl, Steffen Limmer

2024

Preprint:

Copyright ACM 2024. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in GECCO '24: Proceedings of the Genetic and Evolutionary Computation Conference, https://doi.org/10.1145/3638529.3654057.

Letting a Large Neighborhood Search for an Electric Dial-A-Ride Problem Fly: On-The-Fly Charging Station Insertion

Maria Bresich*
Institute of Logic and Computation, TU Wien
Vienna, Austria
mbresich@ac.tuwien.ac.at

Günther R. Raidl
Institute of Logic and Computation, TU Wien
Vienna, Austria
raidl@ac.tuwien.ac.at

Steffen Limmer
Honda Research Institute Europe GmbH
Offenbach/Main, Germany
steffen.limmer@honda-ri.de

ABSTRACT

We consider the electric autonomous dial-a-ride problem (E-ADARP), a challenging extension of the dial-a-ride problem with the goal of finding minimum cost routes serving given transportation requests with a fleet of electric and autonomous vehicles (EAVs). Special emphasis lies on the minimization of user excess ride time under consideration of the charging requirements of the EAVs while constraints regarding, for example, user ride times and time windows have to be satisfied. We propose a novel large neighborhood search (LNS) approach for the E-ADARP employing the concept of battery-restricted fragments for route representation and efficient cost computations. For the charging of the EAVs, the scheduling, and route evaluation, we introduce two approaches where one deals with these challenges separately and one provides a combined approach. The first approach uses dedicated LNS operators and a forward labeling algorithm whereas the latter employs a novel route evaluation procedure for inserting charging stops on-the-fly as needed. The performance of our LNS-based algorithms is evaluated on common benchmark instances and results show that especially the approach with the on-the-fly insertion almost consistently outperforms former state-of-the-art techniques, finding new best-known solutions for many instances.

CCS CONCEPTS

• **Theory of computation** → *Random search heuristics*; **Randomized local search**; • **Mathematics of computing** → **Combinatorial optimization**; • **Applied computing** → *Transportation.*

KEYWORDS

Dial-A-Ride Problem, Electric Autonomous Vehicles, Large Neighborhood Search

ACM Reference Format:
Maria Bresich, Günther R. Raidl, and Steffen Limmer. 2024. Letting a Large Neighborhood Search for an Electric Dial-A-Ride Problem Fly: On-The-Fly Charging Station Insertion. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638529.3654057

*We acknowledge the financial support from Honda Research Institute Europe GmbH.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0494-9/24/07
https://doi.org/10.1145/3638529.3654057

1 INTRODUCTION

Increasing mobility and transportation demands especially in urban areas lead to a growing interest in on-demand transportation as well as ride-sharing services [35]. Such services are gaining popularity as they provide more flexibility than public transportation with its fixed schedules, and sharing rides with other users usually decreases the costs and environmental impact compared to privately owned cars and classic taxi services. They fall into the class of dial-a-ride (DAR) systems, which have been around since 1970 [25].

The dial-a-ride problem (DARP) is concerned with the planning and scheduling of such DAR services where a fleet of vehicles is employed to fulfill transportation requests of users. Each request specifies a pickup and a drop-off location as well as the desired time window, when service should be provided at one of these locations. The standard DARP as defined by Cordeau and Laporte [7] aims at finding minimum cost routes serving all requests while satisfying a set of constraints. Typical constraints consider operational aspects including time windows, vehicle capacities, and maximum route durations, as well as user experience by establishing a maximum user ride time. The literature on DARPs encompasses diverse other objectives and constraints, and comprehensive reviews are given by Molenbruch et al. [23] and Ho et al. [14].

These reviews also provide excellent overviews of established solving approaches, most of which can be classified as either exact methods or (meta-)heuristics. Exact algorithms often make use of mixed integer linear programming (MILP) techniques like branch-and-cut [4–6, 26, 33], branch-and-price [11, 27, 28, 32], or branch-and-price-and-cut [13, 30]. Unfortunately, these exact methods often do not scale well enough in practice and therefore diverse (meta-)heuristics have been proposed for DARPs, including tabu search [7], deterministic annealing (DA) [4], large neighborhood search (LNS) [5, 22], and genetic algorithms [8, 16].

Aside from ride-sharing services, another approach to environment-friendlier mobility is the employment of electric vehicles (EVs), whose adoption is quickly progressing. Additionally, the development and introduction of self-driving cars, or autonomous vehicles (AVs), are advancing and offer new opportunities. As AVs do not require human drivers, they provide greater flexibility and more operating time without having to return to a depot. Thus, the provision of ride-sharing services by means of electric autonomous vehicles (EAVs) is a highly promising advancement [24]. The *electric autonomous dial-a-ride problem* (E-ADARP) as introduced by Bongiovanni et al. [3] considers EAVs within the DARP, which requires to deal with the battery management. Employing EVs for a DAR service requires to plan in recharging of the vehicles, which affects the routing as well as the scheduling. Detours to charging stations

and the vehicles' battery levels have to be taken into account when planning the routes, and charging durations must be scheduled in a way that prevents battery depletion as well as time window violations. In the E-ADARP, (partial) recharging en route is allowed whenever no passenger is on board, and a minimum battery level, or state of charge (SOC), for the end of a route is specified. These and other additional constraints further increase the practical complexity of the standard DARP, which is already NP-hard, and call for new solution approaches.

This paper presents a large neighborhood search for the E-ADARP and we propose two approaches to deal with the charging of the vehicles as well as the scheduling and evaluation of routes. The first approach handles these concepts individually by employing dedicated LNS operators to deal with charging stops together with a modified version of the labeling algorithm by Su et al. [37] for the evaluation. For the second approach, we introduce a novel route evaluation procedure that inserts charging stops on-the-fly as needed while also determining the respective charging durations. We formulate the underlying charging station insertion subproblem as a MILP, before proposing a time-efficient heuristic. The most important original features that distinguish our work from the literature are (a) a structurally rather simple LNS framework with only few destroy and repair operators, (b) the consideration of battery-restricted fragments and their caching for efficient route evaluation, and (c) the highly effective on-the-fly insertion of charging stops. Computational experiments are performed on common benchmark instances and results are compared to exact and heuristic approaches from the literature [3, 19, 37]. Results show that especially the approach with the on-the-fly insertion of charging stops almost consistently outperforms all former methods, and for 23 out of 48 instances new best-known solutions were found.

The remainder is organized as follows. The next section reviews related work. A concrete definition of the E-ADARP is provided in Section 3. Section 4 details the LNS framework as well as our two proposed approaches for charging stop insertion and route evaluation. Section 5 describes the experiments and discusses the results. Concluding remarks and an outlook on promising future research lines are given in Section 6.

2 RELATED WORK

So far only few solving approaches have been proposed for the E-ADARP as considered here, which we discuss in the following.

In 2019, Bongiovanni et al. [3] introduced the E-ADARP and model it by three- and two-indexed MILP formulations. The latter is more efficient but less expressive as it cannot handle vehicle-dependent battery consumptions and recharge rates. They further strengthen the models with new valid inequalities and solve them in a branch-and-cut algorithm. A performance evaluation on two sets of new benchmark instances indicates that the two-indexed formulation mostly outperforms the three-indexed one.

Bongiovanni et al. [2] tackle the dynamic version of the E-ADARP by sequentially solving static E-ADARP subproblems with a two-phase metaheuristic. In the first phase, a greedy insertion heuristic assigns requests to vehicles and vehicle schedules are computed. A machine learning-based large neighborhood search is employed in the second phase for re-optimization where routes from the first phase are modified and improved by means of destroy and repair operators, which are selected by an offline trained machine learning model.

In [1], Bongiovanni et al. propose a scheduling algorithm which aims to minimize user ride time and can be applied to DARPs in general. To deal with charging decisions for EVs as used in the E-ADARP, they also introduce a battery management heuristic.

Su et al. [37] proposed a *deterministic annealing* (DA) metaheuristic for the (static) E-ADARP. It employs various local search operators for intra- and inter-route modification. The authors also present the first exact user excess ride time optimization involving a route evaluation scheme of linear time complexity that is based on a forward labeling algorithm. They introduce the notion of *battery-restricted fragments* for a new path representation of E-ADARP routes, which allows for minimization of user excess ride time. Each fragment represents a subsequence of pickup and drop-off locations in between which no charging is possible; for these the user excess ride time can be independently optimized by solving a linear program. To enable an efficient route evaluation, all feasible fragments are enumerated and evaluated in a preprocessing phase. The performance of the DA algorithm is tested on three sets of benchmark instances, two of which coming from [3]. Compared to the MILP-based approaches from [3], the DA algorithm performs better on several instances.

About the same time as Su et al., Limmer [19] proposed a *bilevel large neighborhood search* (BI-LNS). In this approach, an outer level employs variation operators to only schedule the charging of the EVs. In the inner level, the charging stops are considered fixed, and destroy and repair operators are used for optimizing the insertion of pickup and drop-off stops for fulfilling user requests. While the fixation of charging stops in the outer level may appear rather crude and naive, the big advantage of this approach is that it enables a fast and simple realization of the inner level operations. The actual route evaluation can be done in a classical way based on forward time slacks. This improves the overall scalability in comparison to more expensive operators as used, for example, in [37]. Experimental results show that BI-LNS is competitive on the smaller instances from [3] and outperforms DA on the larger instances from [37]. Moreover, also some new very large instances with hundreds of vehicles and thousands of requests are considered, illustrating the excellent scalability of the approach.

3 FORMAL DEFINITIONS

This section first specifies the E-ADARP following the original definition from [3]. Then, the concept of battery-restricted fragments and how they are used for user excess ride time optimization of routes is explained.

The E-ADARP is defined on a complete directed graph $G = (V, A)$ with vertex set V that represents all relevant geographic locations and arc set $A = \{(i, j) : i, j \in V, i \neq j\}$. A fleet of n_K EAVs, denoted by $K = \{1, \ldots, n_k\}$, is considered for providing service for n customer requests within a planning horizon of T^{plan} time units. The set $V = N \cup O \cup F \cup S$ is composed of multiple subsets, each denoting a different type of location. Let $N = P \cup D$ be the set of all customer locations, where $P = \{1, \ldots, n\}$ denotes the pickup locations and $D = \{n + 1, \ldots, 2n\}$ denotes the drop-off

locations. Thus, a pair $(i, i+n)$ represents the i-th request with pickup location $i \in P$ and drop-off location $i+n \in D$; moreover, a maximum user ride time u_i is given for each request $i \in P$. As in [19], each vehicle $k \in K$ is associated with an origin depot location $o_k \in O$ and a destination depot location $f_k \in F$, with $|O| = |F| = n_K$. Note that [3] allows vehicles to select from multiple destination depots. However, on the benchmark instances used here including those adopted from [3], this makes no difference since all of them actually consider only a unique destination depot location. Finally, the set S denotes the available charging stations (CSs), and each station $s \in S$ has a charging rate α_s, which specifies the amount of energy charged per time unit. Vehicles are allowed to either partially or fully recharge when visiting a charging station.

Each vertex $i \in V$ is associated with a time window $[w_i^{\text{start}}, w_i^{\text{end}}]$ with w_i^{start} being the earliest possible and w_i^{end} being the latest possible service start time. Service at a pickup or drop-off location $i \in N$ is assumed to take a certain duration $d_i \geq 0$, whereas $d_i = 0$ for other locations $i \in V \setminus N$. When a vehicle serves a location $i \in V$, its load changes by an amount l_i depending on the number of customers to be picked up or dropped off; i.e., $l_i > 0$ at pickup locations $i \in P$, $l_i = -l_{i-n}$ at drop-off locations $i \in D$, and $l_i = 0$ at all other locations $i \in V \setminus N$. Vehicles $k \in K$ are heterogeneous in terms of their maximum (load) capacities C_k and homogeneous in terms of their battery capacities Q. The initial battery level of a vehicle $k \in K$ at its origin depot is given as $B_{k,1}$. A minimum battery level ratio $\gamma \in [0, 1]$ determines the minimum battery level γQ that a vehicle needs to satisfy when arriving at its destination depot f_k.

Each arc $(i, j) \in A$ is associated with a travel time $t_{i,j}$ and a battery consumption $\beta_{i,j}$. We assume that the triangle inequality holds for travel times as well as battery consumptions.

A path in graph G starting at the respective origin depot, potentially passing through pickup, drop-off, and charging station locations, and ending at a destination depot is called the *route* of a vehicle. An E-ADARP solution is then composed of n_K vehicle routes together with their respective schedules, specifying the service start time t_i^{serv} at each location/node i of the route.

3.1 Further Constraints

A solution of the E-ADARP is feasible if its routes satisfy the following further constraints.

- All requests are served exactly once, meaning that there is for each request $i = 1, \ldots, n$ one route that visits pickup location i and drop-off location $i+n$ in this order.
- The maximum EAV capacity C_k is not exceeded at any time.
- The maximum user ride time u_i of each request is respected.
- Service starts at each location $i \in V$ within its time window $[w_i^{\text{start}}, w_i^{\text{end}}]$. While it is not allowed to arrive later than w_i^{end}, vehicles may arrive earlier than w_i^{start}, which induces some waiting time.
- Each vehicle k starts its route with the initial battery levels $B_{k,1}$, may never run out of energy or be charged beyond its battery capacity, and has to meet the minimum required battery level γQ when reaching its destination depot.
- Only EAVs not carrying a load may visit charging stations.
- Over all routes, each charging station can only be visited at most once. Note that this constraint can also be relaxed or removed to allow multiple visits of each CS, as considered in [3, 19, 37].

In contrast to the standard DARP [7], there are no constraints limiting the total duration of routes as there are no human drivers when employing AVs.

3.2 Objective

The goal of the E-ADARP is to find a feasible solution that minimizes as costs a weighted sum of the total travel time over all routes and the total user excess ride time over all requests $i \in P$, i.e.,

$$\min w^{\text{routing}} \sum_{k \in K} \sum_{(i,j) \in A} t_{i,j} x_{i,j}^k + w^{\text{excess}} \sum_{i \in P} t_i^{\text{excess}} \tag{1}$$

where w^{routing} and w^{excess} are the weight factors for the two objectives and $x_{i,j}^k$ is a binary decision variable indicating whether vehicle k travels from location i directly to j. The user excess ride time t_i^{excess} of request $i \in P$ is calculated as the difference between the actual ride time of i and the direct travel time $t_{i,i+n}$ from pickup to drop-off, i.e., $t_i^{\text{excess}} = t_{i+n}^{\text{serv}} - t_i^{\text{serv}} - d_i - t_{i,i+n}$.

3.3 Battery-Restricted Fragments

The consideration of user excess ride time in the objective function is an important additional feature of the E-ADARP compared to the standard DARP. Appropriately dealing with this aspect in the scheduling and evaluation of routes is thus crucial. To this end, Su et al. [37] introduce the concept of a battery-restricted fragment, which is an extension of the fragments from Rist and Forbes [31] for the DARP by also taking battery-related matters into account. A battery-restricted fragment $\mathcal{F}$ is defined as a subsequence of a feasible route consisting only of pickup and drop-off locations such that the vehicle's load is zero on arrival at the start and on departure at the end location and positive at all other stops. A route can then be represented as a combination of battery-restricted fragments and charging stations, embedded within the origin and destination depots. Let $P_{\mathcal{F}}$ be the set of pickup locations in a battery-restricted fragment $\mathcal{F}$, the user excess ride time of $\mathcal{F}$ can be minimized by solving the following linear program (LP)

$$\min \sum_{i \in P_{\mathcal{F}}} (t_{i+n}^{\text{serv}} - t_i^{\text{serv}} - d_i - t_{i,i+n}) \tag{2}$$

s.t.

$$t_i^{\text{serv}} + t_{i,j} + d_i \leq t_j^{\text{serv}} \qquad \forall i \in \mathcal{F},\ idx_i \neq |\mathcal{F}|,\ idx_j = idx_i + 1 \tag{3}$$

$$t_{i+n}^{\text{serv}} - t_i^{\text{serv}} - d_i \leq u_i \qquad \forall i \in P_{\mathcal{F}} \tag{4}$$

$$w_i^{\text{start}} \leq t_i^{\text{serv}} \leq w_i^{\text{end}} \qquad \forall i \in \mathcal{F} \tag{5}$$

with idx_i denoting the index of location i in $\mathcal{F}$. The minimum user excess ride time of a route is then equal to the sum of minimum user excess ride times of all contained battery-restricted fragments (simply called "fragments" from now on) [37]. In [37], this property is exploited by enumerating and evaluating all feasible fragments in a preprocessing phase and then using the precalculated user excess ride times during route evaluation when needed.

This approach, however, may be inefficient for instances with a large number of requests, as possibly also fragments are evaluated and stored that never appear in any candidate solution of the metaheuristic. For very large instances, this preprocessing may also already be too time-consuming or too memory demanding.

In our work, we refrain from initially enumerating all possible fragments. Instead, we evaluate fragments only at their first encounter during the LNS and store them with their minimum user

excess ride times in a cache realized by a hash table. For the benchmark instances used in our experiments, the memory consumption of this cache never exceeded 3 GB. Thus, it was not necessary to discard least recently used entries, which would be the solution when some memory limit is reached when dealing with larger instances.

4 LARGE NEIGHBORHOOD SEARCH

Our LNS framework follows a rather simple and classical destroy and repair scheme similar to [29, 36]: Generally, a destroy operator determines which of the problem's decision variables are to be re-optimized while leaving the remaining variables fixed to their values in the incumbent solution. The repair method then tries to restore a valid solution by heuristically solving the induced subproblem. In the standard DARP, this is realized by removing some requests with their stops from the tours and then reinserting them again by some heuristics. Not only the DARP [5, 21, 22] and the E-ADARP [19] have been successfully solved by LNS approaches but also a multitude of other challenging combinatorial optimization problems including routing, scheduling, and location problems [15, 36].

In contrast to many former LNS approaches, we utilize only few destroy and repair operators, do the selection of the destroy and repair operators to be applied simply in a uniform random way without any adaption of selection probabilities, and also always only accept improved solutions as new incumbents. However, we have to face two specific challenges of the E-ADARP: (a) the inclusion of appropriate charging stops and (b) the route evaluation including the determination of best service start and possible waiting times. We propose two ways to deal with these issues where the first one handles them separately and the second one provides a combined approach. In the first approach, called *direct* approach, the LNS is directly concerned with the charging stops and the optimization of their insertion into routes by employing dedicated destroy and repair operators. The used insertion heuristic can also be applied in combination with a repair operator for requests in order to provide more and better insertion possibilities. For the evaluation, we then use a modified version of the labeling algorithm from [37]. The combined and more indirect approach does not involve additional LNS operators but introduces a novel route evaluation procedure that inserts charging stops *on-the-fly* as needed, hereby also determining the respective charging times. In the following, we first describe the main components of the LNS such as the generation of an initial solution and the employed destroy and repair operators for requests. Then, we provide more details on our proposed approaches including a description of the operators for CSs, the labeling algorithm, as well as the MILP formulation and heuristic for the on-the-fly insertion.

4.1 Initial Solution Construction

An initial solution is constructed by starting for each vehicle with an empty route leading from the vehicle's origin depot directly to its destination depot. If our direct approach is employed, an available charging station nearest to the respective origin depot is next inserted into any route. Then, it is tried to insert all customer requests into the routes by the time window order based repair operator described below. As this is a heuristic, not all requests can always be inserted, i.e., the resulting solution may be infeasible in this respect. The following LNS tries to resolve this and come up with a solution covering all requests as quickly as possible. For this, infeasible solutions are always considered to be worse than feasible ones, and among two infeasible solutions one that satisfies more requests is considered better. Ties are broken randomly.

4.2 Destroy and Repair Operators for Requests

For the deletion of customer requests from routes, we employ the popular random removal [34] where κ_r served requests are selected uniformly at random from the routes and removed, with κ_r being the degree of destruction. For repair, we employ three different methods. (a) The greedy heuristic from [34] that we adopt tries to insert the pickup and drop-off stops of every unserved request at every possible position in each route and realizes the overall cheapest insertion. This process is repeated until either all requests are served or no further feasible request insertion is possible. To speed up the process, we cache the insertion cost for each request-route combination in the first iteration and only recalculate it for modified routes in the following steps. (b) Our second repair method works similarly but reduces the computational effort by first sorting the unserved requests in non-decreasing order of the start of their pickup time windows. Then, these requests are tried for insertion in this order. (c) Finally, in our third repair method, the unserved requests are considered for greedy insertion in a random order to reduce the danger of getting trapped in a local optimum. Similarly to [19, 34], random noise is added to the increases of the objective function values associated with the different position combinations when searching for a request's cheapest insertion for further diversification. The noise is randomly sampled from $[-\eta t^{\max}, \eta t^{\max}]$ with $t^{\max} = \max_{i,j \in V} t_{i,j}$, where the noise rate η is a parameter regulating the amount of noise used.

4.3 Direct Approach

In this approach, we have dedicated LNS operators for removing and reinserting charging stations. Similar approaches have been proposed for electric vehicle routing problem (EVRP) variants in [17, 18]. In contrast to these, we use a labeling algorithm for solution evaluation during repair.

Destroy and Repair Operators for Charging Stations. Similarly to the destroy operator for requests, the destroy operator for charging stations selects κ_s visited charging stations uniformly at random and removes them from the routes. In the repair operator, we first have to determine in which state each vehicle route is after the destroy. If a route is still feasible, either because it has not been modified or because the removed charging station(s) were actually redundant, it is left unchanged. If a route now violates time window constraints due to potentially necessary longer charging durations at remaining CSs in the route or if it is battery infeasible and there are no available charging stations left, feasibility is restored by removing as many stops as necessary, starting from the first visited location after the origin depot. Otherwise the route violates battery constraints with at least one charging station available, and in this case, the following greedy heuristic is applied. It identifies the first customer in the route at which the vehicle arrives with negative battery level and also which fragment this customer belongs to. Then, an available charging station which increases the route distance the least is

identified and inserted on the arc between this and the preceding fragment. If there is no feasible charging station or if the insertion leads to time window violations, the step is reverted and an insertion before the previous fragments is tried in the same manner. Similarly, if the route is still not battery feasible after a successful insertion, the procedure is repeated considering the whole route again. In case of remaining constraint violations but no feasible insertion possibility, the heuristic reverts to removing stops from the route.

In the direct approach, this heuristic is also employed within the repair operators for requests. Whenever a candidate route violates only battery-related constraints after request insertion, charging stations are greedily inserted to potentially restore the route's feasibility, leading to an increased number of feasible request insertion possibilities.

Labeling Algorithm. The labeling algorithm we use for evaluating candidate routes is a modified version of the ones from [37] for the E-ADARP and [9] for the EVRP with time windows. Thus, we refer the interested reader to these works for a detailed description and only present the main idea and our adjustments in the following. In the algorithm, each node in a route is assigned a label composed of four resource attributes. The label for the first stop, the origin depot, is defined and the labels of the succeeding nodes are computed one by one using resource extension functions. In the end, the label of each stop has to satisfy certain conditions for the route to be feasible. This procedure runs in linear time with regards to the length of the considered route. As charging stops can only be inserted between fragments but not within, and each fragment can be independently optimized and results are cached, cf. LP (2–5) and Section 3.3, we can reduce the computational effort by assigning labels to fragments instead of individual stops. A route stop is then either a fragment, a depot, or a charging station. Hence, we consider not only the battery consumption for traveling in between two stops but also the battery requirements b_i of the current stop i when computing the labels. The battery requirement of a fragment is the sum of battery consumptions for traveling all the arcs within the fragment. For origin depots and charging stations, b_i is set to zero and we use the minimum battery level at destination depots. The latter is necessary to prevent incorrect infeasibility declarations that can occur with the labeling algorithm in [37]. These are due to the algorithm's goal of charging as much as possible at visited CSs while still arriving as early as possible within the time window of each location, which can lead to zero charging time being scheduled. Another modification concerns the initial label for the origin depot, for which we additionally consider the respective time window as well as the possibility of the vehicle not being fully charged.

An interesting property of the labeling algorithm is that the route evaluation can be done without calculation of service start and charging times, which are not needed when pairing the evaluation with the fragment-based optimization of the user excess ride times as charging costs are not considered in the objective function. However, for the final solution, it is possible to extract necessary charging durations from the labeling process and together with the inner-fragment waiting times derived from the LP, service start times can be determined in a single linear-time forward pass.

4.4 On-The-Fly (OTF) Approach

Contrary to the direct approach, the LNS framework is here not concerned with charging station visits at all. Instead, we introduce a novel route evaluation procedure that inserts such charging stops on-the-fly as needed while also computing the respective charging and service start times. The actual route and its length are here only known after applying this evaluation algorithm. The algorithm is developed with the following assumptions in mind. (a) At most one charging stop needs to be inserted between any pair of successive fragments or depots, which is in contrast to the other approach where there is no such limitation. For practical applications in urban areas, this will be no serious restriction. (b) For the sake of simplicity, we further assume that charging stations have no time windows, although considering them would be straightforward. One major advantage of the OTF approach is its ability to handle vehicle/station-individual charging speeds, which is not possible with the labeling algorithm. In the following, we first consider a MILP formulation of the charging station insertion and evaluation subproblem before we propose a time-efficient heuristic procedure.

MILP Formulation. For a route r, R represents its abstract equivalent where sequences of pickup and drop-off locations are replaced by corresponding fragments. Let $R = R_1, \ldots, R_{n_R}$ denote the abstract route after removal of all possibly so far existing charging stops and B_1 the vehicle's initial battery level when leaving the origin depot. In the following, we abuse notations such as $t_{i,j}$ and d_i by using indices i and j into the current route R instead of the actually corresponding locations. Binary decision variables $x_{i,s}$ for $i = 2, \ldots, |R|$ indicate with value one that a charging stop is included at station s between stops $i-1$ and i of the route. If a charging station s is visited before the i-th stop, the charging duration is assigned to $d^{\text{ch}}_{i,s}$, otherwise it is set to zero. The battery level on arrival at the i-th stop of the route is given by B_i, whereas B^{ch}_i represents the battery level on arrival at the charging station before the i-th stop; if there is no charging station visit before the i-th stop, $B^{\text{ch}}_i = B_{i-1}$. Variables t^{serv}_i denote the time at which service starts at the i-th stop and in case of a charging station, this is the start of the charging duration. The MILP model is then defined as follows.

$$\min \quad \sum_{i=2}^{|R|} \sum_{s \in S} (t_{i-1,s} + t_{s,i} - t_{i-1,i}) \cdot x_{i,s} \tag{6}$$

$$\text{s.t.} \quad \sum_{s \in S} x_{i,s} \leq 1 \qquad i = 2, \ldots, |R| \tag{7}$$

$$t^{\text{serv}}_i \geq t^{\text{serv}}_{i-1} + d_{i-1} + t_{i-1,i} + \sum_{s \in S} \left((t_{i-1,s} + t_{s,i} - t_{i-1,i}) \cdot x_{i,s} + d^{\text{ch}}_{i,s} \right) \qquad i = 2, \ldots, |R| \tag{8}$$

$$B^{\text{ch}}_i = B_{i-1} - \sum_{s \in S} \beta_{i-1,s} x_{i,s} \qquad i = 2, \ldots, |R| \tag{9}$$

$$B^{\text{ch}}_i + \sum_{s \in S} \alpha_s d^{\text{ch}}_{i,s} \leq Q \qquad i = 2, \ldots, |R| \tag{10}$$

$$B_i = B^{\text{ch}}_i + \sum_{s \in S} \alpha_s d^{\text{ch}}_{i,s} - \beta_{i-1,i} - \sum_{s \in S} (\beta_{s,i} - \beta_{i-1,i}) \cdot x_{i,s} - b_i \qquad i = 2, \ldots, |R| \tag{11}$$

$$d^{\text{ch}}_{i,s} \leq \frac{Q}{\alpha_s} x_{i,s} \qquad i = 2, \ldots, |R|,\ s \in S \tag{12}$$

$$x_{i,s} \in \{0, 1\} \qquad i = 2, \ldots, |R|,\ s \in S \tag{13}$$

$$d^{\text{ch}}_{i,s} \geq 0 \qquad i = 2, \ldots, |R|,\ s \in S \tag{14}$$

$$0 \leq B_i \leq Q \qquad i = 2, \ldots, |R| \tag{15}$$

$$0 \leq B^{\text{ch}}_i \leq Q \qquad i = 2, \ldots, |R| \tag{16}$$

$$w^{\text{start}}_i \leq t^{\text{serv}}_i \leq w^{\text{stop}}_i \qquad i = 1, \ldots, |R| \tag{17}$$

The objective function (6) minimizes the total detour length caused by all newly inserted charging visits in order to increase the objective value of the route as little as necessary. Inequalities (7) limit the number of charging visits between successive stops to one, and Constraints (8) ensure feasible service start times. Equations (9) set the battery levels on arrival at a charging station, whereas Inequalities (11) determine the battery levels on arrival at all other route stops taking a potential preceding charging visit into account. Constraints (10) prevent overcharging of vehicles, and Constraints (12) restrict the charging durations. Lastly, (13)–(17) state the domains of the decision variables. Solving this MILP model for a route tells whether the route is feasible or not as well as where and how long the vehicle has to charge and when service starts at all stops.

Time-Efficient Heuristic. In the LNS, many candidate routes have to be evaluated, and the above MILP approach can therefore be expected to be a serious bottleneck. Therefore, we propose as an alternative a fast heuristic procedure for simultaneously evaluating a route and inserting charging stations on-the-fly, where we again start with the fragment-based representation R of the route after removal of all charging stops.

Algorithm 1: OTF charging stop insertion and evaluation.

Input: route r
Output: feasibility status of r
$r \leftarrow$ remove_charging_stops(r);
$R \leftarrow$ get_fragments_and_depots(r);
if contains_infeasible_fragment(R) **then**
  return false;
$i^{\text{ch}} \leftarrow none$, $S' \leftarrow$ available_stations(S);
t^{early}, d^{wait}, B, $i^{\text{ch}} \leftarrow$ forward_pass(R);
t^{late}, $d^{\text{wait_back}} \leftarrow$ backward_pass(R);
if violates_time_windows(t^{early}) **or** violates_time_windows(t^{late}) **then**
  return false;
while $i^{\text{ch}} \neq none$ **do**
  $min_detour \leftarrow \infty$, $max_energy \leftarrow 0$, $\hat{i} \leftarrow none$;
  for i in $i^{\text{ch}}, \dots, 2$, $s \in S'$ **do**
    $feasible$, $detour$, $energy \leftarrow$ check_stop(R, i, s);
    if $feasible$ **and** better_stop($detour, energy, min_detour, max_energy$) **then**
      $min_detour \leftarrow detour$, $max_energy \leftarrow energy$;
      $\hat{i} \leftarrow i$, $\hat{s} \leftarrow s$;
  if $\hat{i} = none$ **then**
    return false;
  insert_charging_stop($r, R, \hat{i}, \hat{s}$);
  update($t^{\text{early}}, d^{\text{wait}}, t^{\text{late}}, d^{\text{wait_back}}, B, i^{\text{ch}}, S'$);
return true;

Algorithm 1 shows the main steps of this procedure in pseudocode. In a forward-pass on R, we determine for all stops $i \in R$ the earliest possible service start time t_i^{early}, waiting duration d_i^{wait} (which is induced by a vehicle's arrival before the start of the time window), and battery level B_i on arrival. Similarly, the latest possible service start times t_i^{late} as well as "backward-waiting" durations $d_i^{\text{wait_back}}$ when waiting for the end of time windows are determined in a backward-pass on R. If any time window violation is detected at this step, the route is already proven infeasible and the procedure stops. Charging is necessary if at least one B_i assumes a negative value; in this case, we identify the latest stop i^{ch} before which this value gets negative. Otherwise, if all B_i are non-negative and no charging is required, the route is declared feasible as it is and the algorithm returns. For the insertion of a charging stop, we consider all stops $i \in \{2, \dots, i^{\text{ch}}\}$ in reverse order and all available charging stations $s \in S'$ and check for all combinations whether a charging stop at s before i would be feasible and how much energy could be charged. If no feasible charging stop is found, the route's feasibility cannot be restored and we can stop. Otherwise, from all feasible insertion options, we select the one that introduces the smallest detour length and provides a long enough charging duration to satisfy all battery requirements. If no such charging stop exists, the one with the highest amount of charged energy after deduction of the battery consumption required for the incurred detour is chosen. The selected charging stop is then inserted into the route and S' as well as t_i^{early}, t_i^{late}, d_i^{wait}, $d_i^{\text{wait_back}}$, and B_i are updated for all $i \in R$ affected by the insertion. We determine again whether and where charging is necessary at the latest and repeat the required steps for finding, selecting, and inserting a best suited charging stop until the route becomes feasible or the procedure terminates due to the non-existence of further feasible insertion options.

A property of this algorithm is that when feasibility of a route can be restored by a single charging stop insertion, it will do this in an optimal way. Otherwise, it will still always find suitable charging station insertions if charging rates are homogeneous and there is a way to make the solution feasible, but optimality in respect to the tour length is not guaranteed anymore. Excluding the evaluation of fragments appearing the first time via the LP, the algorithm runs in time $O(|R| \cdot |S'| \cdot n_{\text{ch}})$ with n_{ch} being the number of inserted charging stops. Considering existing benchmark instances, the number of available CSs $|S'|$ is at most three and only very few charging stops are typically required per route (≤ 3), and we may therefore neglect these factors and say the algorithm runs in amortized linear time in the number of stops.

5 EXPERIMENTAL ANALYSIS

All experiments were performed on single cores of 2.4 GHz Intel Xeon E5-2640 v4 processors and with a memory limit of 20 GB. Our proposed approaches are implemented in Julia 1.10.0, and we used Gurobi 10.0.3[1] as MILP solver in single-threaded mode. We consider two sets of benchmark instances based on instances for the standard DARP: 14 Cordeau instances[2] [6] and ten Ropke instances[3] [33]. All instances are enhanced with E-ADARP-specific features as detailed in [3], e.g., by adding three charging stations; and they are named using the pattern an_K-n, where n_K is the number of vehicles and n the number of requests. We do not consider very large instances such as in [19], on which the detailed optimization of charging visits that we focus on turned out to be too time-consuming. Like in [3, 19, 37], $w^{\text{routing}} = 0.75$ and $w^{\text{excess}} = 0.25$ are used as weight factors in the objective function (1). The employed battery consumption $\beta_{i,j}$ is proportional to the travel time $t_{i,j}$. We consider a minimum battery level ratio $\gamma = 0.7$ of a vehicle's battery capacity required on arrival at the destination depot. Besides, we investigate two modes for charging station visits: a limitation to one visit per station over all

[1] https://www.gurobi.com
[2] https://luts.epfl.ch/wpcontent/uploads/2019/03/e_ADARP_archive.zip
[3] Available under https://github.com/HRI-EU/e_adarp_material

Table 1: Results on Cordeau instances with $\gamma = 0.7$ for limited CS visits ($n_s = 1$) and unlimited CS visits ($n_s = \infty$).

Instance	e-ADARP2 (Bongiovanni et al., 2019)		DA (Su et al., 2023)			BI-LNS (Limmer, 2023)			Direct Approach				OTF Approach MILP				OTF Approach Heuristic			
	RT [min]	Obj	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Feas	Obj min	Obj mean	Obj sd	Feas	Obj min	Obj mean	Obj sd	Feas	Obj min	Obj mean	Obj sd	Feas
$n_s = 1$																				
a2-16	0.09	**240.66**	1.60	**240.66**	240.66	242.83	245.50	10/10	**240.66**	242.19	1.03	30/30	**240.66**	240.84	0.40	30/30	**240.66**	240.66	0.00	30/30
a2-20	120.00	NA	2.88	**293.27**	294.11	NA	NA	0/10	**293.27**	297.60	6.44	30/30	**293.27**	294.98	3.30	22/30	**293.27**	293.27	0.00	30/30
a2-24	16.02	358.21	3.44	**353.18**	NA	356.99	363.04	10/10	**353.18**	358.85	5.43	30/30	**353.18**	353.45	1.45	30/30	**353.18**	353.18	0.00	30/30
a3-18	0.80	**240.58**	0.97	**240.58**	240.58	242.49	246.13	10/10	244.35	244.39	0.08	30/30	**240.58**	240.98	0.95	30/30	**240.58**	240.58	0.00	30/30
a3-24	2.54	277.72	2.06	**275.97**	277.43	277.52	277.52	10/10	277.43	278.42	1.18	30/30	**275.97**	277.46	1.76	30/30	**275.97**	275.97	0.00	30/30
a3-30	120.00	NA	1.30	**424.93**	436.20	432.27	436.56	10/10	**424.93**	430.07	4.05	30/30	**424.93**	430.32	3.59	30/30	**424.93**	426.12	1.59	30/30
a3-36	120.00	**494.04**	2.09	**494.04**	502.27	496.75	500.84	10/10	**494.04**	497.86	3.83	30/30	**494.04**	502.35	4.27	30/30	**494.04**	497.18	2.61	30/30
a4-16	1.12	**223.13**	0.52	**223.13**	223.13	**223.13**	223.95	10/10	**223.13**	223.13	0.00	30/30	**223.13**	223.37	0.27	30/30	**223.13**	223.13	0.00	30/30
a4-24	30.58	318.21	0.90	**316.65**	318.31	319.37	321.10	10/10	318.31	319.89	0.67	30/30	**316.65**	319.17	1.88	30/30	**316.65**	316.65	0.00	30/30
a4-32	120.00	430.07	1.19	**397.87**	405.85	401.97	402.59	10/10	**397.87**	399.26	2.18	30/30	**397.87**	399.98	2.35	30/30	**397.87**	397.87	0.00	30/30
a4-40	120.00	NA	1.91	479.02	NA	471.72	478.93	10/10	**467.72**	480.89	7.48	30/30	471.78	487.42	7.65	30/30	**467.72**	474.47	6.23	30/30
a4-48	120.00	NA	2.74	582.22	NA	579.71	588.48	10/10	**575.35**	582.71	4.80	30/30	580.49	591.36	6.69	29/30	575.62	579.63	2.40	30/30
a5-40	120.00	447.63	1.63	424.26	436.94	420.20	423.88	10/10	420.01	425.32	3.20	30/30	425.93	433.02	5.60	30/30	**418.75**	421.16	3.26	30/30
a5-50	120.00	NA	2.64	603.24	NA	593.71	602.30	9/10	**589.61**	597.55	5.14	30/30	596.54	612.00	9.81	27/30	**589.61**	596.09	3.87	30/30
$n_s = \infty$																				
a2-16			1.99	**240.66**	240.66	242.44	242.44	10/10	241.32	241.34	0.06	30/30	**240.66**	240.66	0.00	30/30	**240.66**	240.66	0.00	30/30
a2-20			5.27	286.32	288.89	290.33	291.23	10/10	288.39	290.50	1.14	30/30	**285.86**	285.95	0.21	30/30	**285.86**	285.86	0.00	30/30
a2-24			5.96	354.38	374.68	354.53	356.89	10/10	**350.32**	355.40	3.02	30/30	**350.32**	350.33	0.05	30/30	**350.32**	350.32	0.00	30/30
a3-18			1.10	**238.82**	238.82	241.95	242.46	10/10	241.13	241.13	0.00	30/30	**238.82**	239.55	0.76	30/30	240.03	240.03	0.00	30/30
a3-24			2.50	**275.20**	275.20	277.52	278.02	10/10	276.27	276.62	0.58	30/30	**275.20**	276.05	0.96	30/30	**275.20**	275.20	0.00	30/30
a3-30			2.85	415.71	417.07	419.16	426.30	10/10	416.89	421.54	3.55	30/30	**413.45**	415.85	1.87	30/30	**413.45**	414.08	1.07	30/30
a3-36			5.72	484.85	487.91	490.26	492.79	10/10	484.07	492.96	4.49	30/30	**483.08**	490.60	3.64	30/30	484.49	486.98	3.11	30/30
a4-16			0.52	**222.49**	222.49	**222.49**	223.57	10/10	**222.49**	222.49	0.00	30/30	**222.49**	223.09	0.54	30/30	**222.49**	222.49	0.00	30/30
a4-24			1.18	315.98	317.99	316.51	318.38	10/10	**315.40**	316.48	1.21	30/30	**315.40**	316.00	0.47	30/30	315.98	315.98	0.00	30/30
a4-32			2.06	**394.94**	401.82	396.64	397.98	10/10	**394.94**	394.94	0.00	30/30	**394.94**	395.84	1.19	30/30	**394.94**	394.94	0.00	30/30
a4-40			3.77	458.52	467.60	461.16	461.91	10/10	458.63	462.73	3.34	30/30	**457.76**	465.13	3.85	30/30	457.88	458.67	1.20	30/30
a4-48			6.72	568.08	575.96	568.01	570.80	10/10	565.04	569.97	3.00	30/30	**561.15**	565.94	3.26	30/30	561.38	564.54	1.85	30/30
a5-40			2.50	419.33	425.29	418.79	421.06	10/10	415.96	417.24	1.68	30/30	**415.88**	425.07	3.79	30/30	**415.88**	416.43	1.05	30/30
a5-50			5.88	579.15	588.98	571.37	575.49	10/10	571.13	576.96	4.07	30/30	573.37	586.47	7.02	30/30	**567.61**	571.48	2.62	30/30

routes as in [3], denoted by $n_s = 1$, and an unlimited number of allowed visits as in [19, 37] with $n_s = \infty$. For each instance and configuration, we perform 30 runs of our solving approaches with a time limit of 300 s per run.

To reduce the complexity of the instances, we employ time window tightening and arc elimination as proposed by Dumas et al. [10] and Cordeau [6] for other problems. For the former, we use the rules for the E-ADARP as stated in [37] with some modifications. Regarding arc elimination, we adjust the rules from [12] to consider multiple depots and add a rule based on vehicle loads. The application of further path-based elimination rules [6, 10] allows not only for arc removal but also identification of incompatible request pairs that can never be served by the same route. We exploit this information to reduce the number of insertion positions of a request that have to be checked for feasibility in the LNS repair operators described in Section 4.2, thereby improving their efficiency. Robust settings for the noise rate η as well as the degrees of destruction κ_r and κ_s are identified through parameter tuning with SMAC3[4] [20] by running 10000 trials on ten representative instances. The resulting configuration further used is $\eta = 0.014$, $\kappa_r = 15$, and $\kappa_s = 2$ for $n_s = 1$ and $\kappa_s = 0.54$ for $n_s = \infty$.

5.1 Results

The results of our approaches are compared to the two-indexed MILP formulation e-ADARP2 from [3], the DA approach from [37], and the BI-LNS from [19] using the data reported in these works.

Cordeau Instances. The results on the smaller Cordeau instances under $\gamma = 0.7$ and for both modes regarding the allowed number of charging station visits are shown in Table 1. We report run times (RT) and objective values for the e-ADARP2 and the mean run times as well as the minimum and mean objective values over 50 trials for DA, where "NA" means that some trials did not find a feasible solution. The minimum and average objective values and the number of runs yielding a feasible solution are shown for BI-LNS, our direct approach, and our OTF approaches employing the MILP formulation and the heuristic. For these, the run time was fixed to five minutes per trial and mean values are computed over all obtained feasible solutions. For our proposed approaches, the standard deviation (sd) over all feasible runs is given as well. Value "NA" denotes that no feasible solution was found and bold values mark best known objective values. For some instances, the supposedly optimal results by e-ADARP2 are not globally optimal due to an error as pointed out in [37] and thus, better objective values can be achieved as shown in Table 1 for five instances.

For $n_s = 1$, the OTF insertion heuristic demonstrates the best performance of our approaches and consistently outperforms the state-of-the-art approaches from the literature, matching all best known objective values so far and providing three new best solutions while having average gaps in between –2.31% (on a3-30) and 0.64% (on a3-36) compared to the approaches from literature. The direct approach obtains three new best objective values as well but performs overall slightly worse with mean gaps in the range from –1.41% (on a3-30) to 1.58% (on a3-18). In contrast to the results from literature, both approaches yield feasible solutions in all runs. This is not achieved by the MILP-based OTF insertion procedure for three instances but it is still competitive on most instances. Regarding the mode $n_s = \infty$, no results are reported for e-ADARP2 but all other approaches obtain better results than with $n_s = 1$ for most instances. Both OTF approaches are now clearly superior to the direct approach as well as DA and BI-LNS in terms of best found

[4] https://github.com/automl/SMAC3

Table 2: Results on Ropke instances with $\gamma = 0.7$ for limited CS visits ($n_s = 1$) and unlimited CS visits ($n_s = \infty$).

Instance	DA (Su et al., 2023)			BI-LNS (Limmer, 2023)			Direct Approach				OTF Approach MILP				OTF Approach Heuristic			
	RT mean [min]	Obj min	Obj mean	Obj min	Obj mean	Feas	Obj min	Obj mean	Obj sd	Feas	Obj min	Obj mean	Obj sd	Feas	Obj min	Obj mean	Obj sd	Feas
$n_s = 1$																		
a5-60	8.46	NA	NA	NA	NA	0/10	NA	NA	NA	0/30	NA	NA	NA	0/30	NA	NA	NA	0/30
a6-48	8.37	NA	NA	519.55	522.50	10/10	517.60	528.24	6.81	30/30	523.49	537.45	9.06	30/30	**517.12**	521.62	2.95	30/30
a6-60	5.45	NA	NA	733.45	742.02	9/10	719.78	737.77	10.75	30/30	736.52	750.14	11.54	10/30	**714.16**	731.45	10.02	30/30
a6-72	9.84	NA	NA	NA	NA	0/10	NA	NA	NA	0/30	NA	NA	NA	0/30	NA	NA	NA	0/30
a7-56	3.69	NA	NA	649.11	669.71	10/10	**633.65**	656.45	12.31	30/30	656.86	675.48	15.31	23/30	636.56	649.37	10.43	30/30
a7-70	8.51	NA	NA	NA	NA	0/10	820.90	840.02	9.75	26/30	NA	NA	NA	0/30	**816.64**	840.59	16.29	30/30
a7-84	13.18	NA	NA	NA	NA	0/10	NA	NA	NA	0/30	NA	NA	NA	0/30	NA	NA	NA	0/30
a8-64	20.12	NA	NA	646.82	652.38	10/10	645.69	656.21	8.63	30/30	652.65	674.25	15.43	30/30	**639.06**	651.33	6.63	30/30
a8-80	14.47	NA	NA	854.85	863.74	10/10	842.23	863.08	10.24	30/30	869.19	888.37	15.02	8/30	**837.79**	862.75	15.14	30/30
a8-96	14.35	NA	NA	NA	NA	0/10	NA	NA	NA	0/30	NA	NA	NA	0/30	NA	NA	NA	0/30
$n_s = \infty$																		
a5-60	8.21	708.54	723.73	697.87	709.11	10/10	693.79	703.04	4.83	30/30	695.81	704.15	5.31	30/30	**686.36**	692.75	4.39	30/30
a6-48	8.07	509.76	525.10	511.04	514.53	10/10	508.22	510.85	2.47	30/30	509.87	519.44	5.22	30/30	**508.10**	509.46	1.88	30/30
a6-60	4.83	697.57	711.52	699.70	705.56	10/10	694.19	700.72	3.36	30/30	699.25	708.04	5.09	30/30	**689.95**	695.13	3.43	30/30
a6-72	9.57	796.19	826.48	788.34	801.80	10/10	780.68	796.83	8.00	30/30	782.32	799.84	11.52	30/30	**769.12**	779.49	4.46	30/30
a7-56	3.53	625.91	641.82	627.34	633.38	10/10	619.63	627.64	4.69	30/30	623.45	632.08	3.91	30/30	**617.12**	621.51	3.02	30/30
a7-70	8.00	781.56	800.35	777.69	785.49	10/10	763.22	773.36	5.76	30/30	769.84	783.03	8.58	30/30	**757.66**	765.36	3.57	30/30
a7-84	11.75	915.61	938.49	900.98	916.93	10/10	892.31	906.41	6.19	30/30	900.77	920.03	9.99	30/30	**888.40**	897.33	6.44	30/30
a8-64	21.50	649.93	668.48	645.62	648.60	10/10	634.36	646.57	5.03	30/30	654.75	663.89	6.35	30/30	**632.95**	641.17	3.77	30/30
a8-80	12.41	843.26	865.90	815.06	825.74	10/10	813.44	823.82	6.09	30/30	821.29	837.30	7.16	30/30	**801.08**	814.87	6.25	30/30
a8-96	13.45	1097.76	1136.43	1072.77	1091.06	10/10	1058.08	1073.35	9.09	30/30	1074.02	1091.18	11.36	30/30	**1048.87**	1060.22	6.23	30/30

solutions. Even though employing the MILP model yields more new best solutions, the heuristic performs better in terms of average objective values and gaps, having mean gaps below zero for all instances but one, i.e., 0.51% for a3-18. This illustrates the overall better performance of the OTF heuristic. The average performance of the direct approach remains almost the same compared to setting $n_s = 1$, with a highest average gap of 1.07% for a3-30.

Ropke Instances. Results for the larger Ropke instances are given in Table 2 in the same way as before. For these instances, e-ADARP2 has not been tested and DA is not able to find any feasible solutions in case of limited CS visits. For this setting and on instances with an average of twelve requests per vehicle, none of the other approaches yield any feasible solutions either. However, the OTF heuristic and the direct approach obtain the first reported feasible results for instance a7-70, with the OTF heuristic being able to solve it in every trial. Regarding the other instances previously solved by BI-LNS, the OTF heuristic consistently achieves better mean objective values with average gaps ranging from −3.04% to −0.12%. The direct approach also exhibits a competitive performance with 1.10% being the highest mean gap, whereas the performance of the MILP-based OTF approach substantially deteriorates for the restricted mode $n_s = 1$ on these larger instances. Like on the Cordeau instances, it performs better when unlimited visits to CSs are allowed but is again inferior to the OTF heuristic and also the direct approach. The latter yields better average objective values than DA and BI-LNS on all instances but is still significantly outperformed by the OTF heuristic. The OTF heuristic dominates by providing not only best mean objective values but also new best objective values for all instances. These results impressively highlight its scalability and ability to optimize charging station visits within a reasonable time.

6 CONCLUSION AND FUTURE WORK

This work proposes a structurally simple LNS framework for the E-ADARP where battery-restricted fragments are evaluated on demand and cached, thus enabling an efficient optimization of user excess ride times. We introduce two approaches to handle the insertion of appropriate charging stops and the route evaluation and scheduling. The OTF approach provides a highly efficient, novel route evaluation procedure that inserts and schedules charging stops on-the-fly by applying either a MILP formulation or a time-efficient heuristic. The direct approach employs dedicated LNS operators for charging stop optimization and an exact, fragment-based labeling algorithm with linear run time for evaluation.

In computational experiments, the performance of the proposed approaches is evaluated on common benchmark instances and compared to exact as well as heuristic approaches from the literature. The direct approach is competitive on smaller instances and beats the established approaches on larger instances, but it is still clearly outperformed by the OTF heuristic, which performs better on almost all instances. The OTF heuristic achieves new best objective values for 23 instances and matches best known solution values for 14 out of 48 instances.

Further improving the scalability of our proposed approaches to very large instances would be an interesting future work, e.g., by adding more sophisticated operators for request optimization. The introduction of machine learning techniques for operator selection would then be another promising extension of our work. Consideration of real-world scenarios including load-dependent charging, more realistic modeling of charging stations with different charging speeds and visitor constraints, as well as the dynamic problem variant constitute further highly practical research lines.

REFERENCES

[1] Claudia Bongiovanni, Nikolas Geroliminis, and Mor Kaspi. 2023. A ride time-oriented scheduling algorithm for dial-a-ride problems. https://doi.org/10.48550/arXiv.2211.07347

[2] Claudia Bongiovanni, Mor Kaspi, Jean-François Cordeau, and Nikolas Geroliminis. 2022. A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations. *Transportation Research Part E: Logistics and Transportation Review* 165 (2022), 102835. https://doi.org/10.1016/j.tre.2022.102835

[3] Claudia Bongiovanni, Mor Kaspi, and Nikolas Geroliminis. 2019. The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological* 122 (2019), 436–456. https://doi.org/10.1016/j.trb.2019.03.004

[4] Kris Braekers, An Caris, and Gerrit K. Janssens. 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological* 67 (2014), 166–186. https://doi.org/10.1016/j.trb.2014.05.007

[5] Kris Braekers and Attila A. Kovacs. 2016. A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological* 94 (2016), 355–377. https://doi.org/10.1016/j.trb.2016.09.010

[6] Jean-François Cordeau. 2006. A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. *Operations Research* 54, 3 (2006), 573–586. https://doi.org/10.1287/opre.1060.0283

[7] Jean-François Cordeau and Gilbert Laporte. 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37, 6 (2003), 579–594. https://doi.org/10.1016/S0191-2615(02)00045-0

[8] Claudio Cubillos, Enrique Urra, and Nibaldo Rodríguez. 2009. Application of Genetic Algorithms for the DARPTW Problem. *International Journal of Computers, Communications & Control* 4, 22 (2009), 127–136. https://doi.org/10.15837/ijccc.2009.2.2420

[9] Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. 2016. Exact Algorithms for Electric Vehicle-Routing Problems with Time Windows. *Operations Research* 64, 6 (2016), 1388–1405. https://doi.org/10.1287/opre.2016.1535

[10] Yvan Dumas, Jacques Desrosiers, and François Soumis. 1991. The pickup and delivery problem with time windows. *European Journal of Operational Research* 54, 1 (1991), 7–22. https://doi.org/10.1016/0377-2217(91)90319-Q

[11] Thierry Garaix, Christian Artigues, Dominique Feillet, and Didier Josselin. 2011. Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. *Computers & Operations Research* 38, 10 (2011), 1435–1442. https://doi.org/10.1016/j.cor.2010.12.014

[12] Dominik Goeke. 2019. Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. *European Journal of Operational Research* 278, 3 (2019), 821–836. https://doi.org/10.1016/j.ejor.2019.05.010

[13] Timo Gschwind and Stefan Irnich. 2015. Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem. *Transportation Science* 49, 2 (2015), 335–354. https://doi.org/10.1287/trsc.2014.0531

[14] Sin C. Ho, Wai Y. Szeto, Yong-Hong Kuo, Janny M. Y. Leung, Matthew Petering, and Terence W. H. Tou. 2018. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological* 111 (2018), 395–421. https://doi.org/10.1016/j.trb.2018.02.001

[15] Thomas Jatschka, Tobias Rodemann, and Günther R. Raidl. 2022. A Large Neighborhood Search for a Cooperative Optimization Approach to Distribute Service Points in Mobility Applications. In *Metaheuristics and Nature Inspired Computing (Communications in Computer and Information Science (CCIS), Vol. 1541)*, Bernabé Dorronsoro, Farouk Yalaoui, El-Ghazali Talbi, and Grégoire Danoy (Eds.). Springer International Publishing, Cham, 3–17. https://doi.org/10.1007/978-3-030-94216-8_1

[16] Rene M. Jorgensen, Jesper Larsen, and Kristin B. Bergvinsdottir. 2007. Solving the Dial-a-Ride problem using genetic algorithms. *Journal of the Operational Research Society* 58, 10 (2007), 1321–1331. https://doi.org/10.1057/palgrave.jors.2602287

[17] Surendra R. Kancharla and Gitakrishnan Ramadurai. 2020. Electric vehicle routing problem with non-linear charging and load-dependent discharging. *Expert Systems with Applications* 160 (2020), 113714. https://doi.org/10.1016/j.eswa.2020.113714

[18] Merve Keskin and Bülent Çatay. 2016. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies* 65 (2016), 111–127. https://doi.org/10.1016/j.trc.2016.01.013

[19] Steffen Limmer. 2023. Bilevel large neighborhood search for the electric autonomous dial-a-ride problem. *Transportation Research Interdisciplinary Perspectives* 21 (2023), 100876. https://doi.org/10.1016/j.trip.2023.100876

[20] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. 2022. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research* 23, 54 (2022), 1–9. http://jmlr.org/papers/v23/21-0888.html

[21] Mohamed A. Masmoudi, Manar Hosny, Kris Braekers, and Abdelaziz Dammak. 2016. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review* 96 (2016), 60–80. https://doi.org/10.1016/j.tre.2016.10.002

[22] Yves Molenbruch, Kris Braekers, and An Caris. 2017. Benefits of horizontal cooperation in dial-a-ride services. *Transportation Research Part E: Logistics and Transportation Review* 107 (2017), 97–119. https://doi.org/10.1016/j.tre.2017.09.001

[23] Yves Molenbruch, Kris Braekers, and An Caris. 2017. Typology and literature review for dial-a-ride problems. *Annals of Operations Research* 259, 1-2 (2017), 295–325. https://doi.org/10.1007/s10479-017-2525-0

[24] Santhanakrishnan Narayanan, Emmanouil Chaniotakis, and Constantinos Antoniou. 2020. Shared autonomous vehicle services: A comprehensive review. *Transportation Research Part C: Emerging Technologies* 111 (2020), 255–293. https://doi.org/10.1016/j.trc.2019.12.008

[25] Philip Oxley. 1980. Dial/a/ride: a review. *Transportation Planning and Technology* 6, 3 (1980), 141–148. https://doi.org/10.1080/03081068008717184

[26] Sophie N. Parragh. 2011. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies* 19, 5 (2011), 912–930. https://doi.org/10.1016/j.trc.2010.06.002

[27] Sophie N. Parragh, Jean-François Cordeau, Karl F. Doerner, and Richard F. Hartl. 2012. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR Spectrum* 34, 3 (2012), 593–633. https://doi.org/10.1007/s00291-010-0229-9

[28] Sophie N. Parragh and Verena Schmid. 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research* 40, 1 (2013), 490–497. https://doi.org/10.1016/j.cor.2012.08.004

[29] David Pisinger and Stefan Ropke. 2010. Large Neighborhood Search. In *Handbook of Metaheuristics*, Michel Gendreau and Jean-Yves Potvin (Eds.). Springer US, Boston, MA, 399–419. https://doi.org/10.1007/978-1-4419-1665-5_13

[30] Yuan Qu and Jonathan F. Bard. 2015. A Branch-and-Price-and-Cut Algorithm for Heterogeneous Pickup and Delivery Problems with Configurable Vehicle Capacity. *Transportation Science* 49, 2 (2015), 254–270. https://doi.org/10.1287/trsc.2014.0524

[31] Yannik Rist and Michael A. Forbes. 2021. A New Formulation for the Dial-a-Ride Problem. *Transportation Science* 55, 5 (2021), 1113–1135. https://doi.org/10.1287/trsc.2021.1044

[32] Stefan Ropke and Jean-François Cordeau. 2009. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science* 43, 3 (2009), 267–286. https://doi.org/10.1287/trsc.1090.0272

[33] Stefan Ropke, Jean-François Cordeau, and Gilbert Laporte. 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks* 49, 4 (2007), 258–272. https://doi.org/10.1002/net.20177

[34] Stefan Ropke and David Pisinger. 2006. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science* 40, 4 (2006), 455–472. https://doi.org/10.1287/trsc.1050.0135

[35] Susan Shaheen and Adam Cohen. 2020. Chapter 3 - Mobility on demand (MOD) and mobility as a service (MaaS): early understanding of shared mobility impacts and public transit partnerships. In *Demand for Emerging Transportation Systems*, Constantinos Antoniou, Dimitrios Efthymiou, and Emmanouil Chaniotakis (Eds.). Elsevier, 37–59. https://doi.org/10.1016/B978-0-12-815018-4.00003-6

[36] Paul Shaw. 1998. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In *Principles and Practice of Constraint Programming*, Michael Maher and Jean-Francois Puget (Eds.). Springer, Heidelberg, 417–431. https://doi.org/10.1007/3-540-49481-2_30

[37] Yue Su, Nicolas Dupin, and Jakob Puchinger. 2023. A deterministic annealing local search for the electric autonomous dial-a-ride problem. *European Journal of Operational Research* 309, 3 (2023), 1091–1111. https://doi.org/10.1016/j.ejor.2023.02.012