Improvements in Large Neighborhood Search for the Electric Autonomous Dial-A-Ride Problem*

Maria Bresich[1(✉)][0009−0000−8291−6765], Günther R. Raidl[1][0000−0002−3293−177X], and Steffen Limmer[2][0000−0003−2385−7886]

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria
{mbresich,raidl}@ac.tuwien.ac.at
[2] Honda Research Institute Europe GmbH, 63073 Offenbach/Main, Germany
steffen.limmer@honda-ri.de

Abstract. We consider a practical extension of the classical dial-a-ride problem (DARP) called the electric autonomous DARP where electric and autonomous vehicles provide service for transportation requests with time windows. The planning and scheduling of routes that minimize not only the vehicles' travel cost but also the user excess ride time while considering charging requirements and operational constraints is a challenging optimization problem. In a previous work, we proposed a large neighborhood search (LNS) with a novel route evaluation approach that heuristically inserts charging stops on-the-fly as needed. Here, we go into more detail regarding the preprocessing procedure for reducing the size of instances as well as the tuning of certain LNS parameters. We further investigate this solving approach by evaluating its performance on different configurations of common benchmark instances, illustrating its successful application throughout. An analysis of the performance and impact of different repair operators provides further insights and reveals improvement opportunities.

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

1 Introduction

One consequence of the growth in urban population are rising mobility demands and associated challenges such as traffic congestion and greenhouse gas emissions. This leads to an increasing interest in on-demand transportation and ride-sharing services as a flexible, affordable, and environment-friendlier alternative not only to privately owned cars but also classic public transportation services. In this context, a transportation request of a user or customer consists of a pickup and a drop-off location together with a service time window for either one. Designing minimum cost tours for a fleet of vehicles to serve a set of such requests is the goal of the dial-a-ride problem (DARP). In the standard DARP [5], the total length of the routes is to be minimized while serving all requests and satisfying

* This project is financially supported by Honda Research Institute Europe GmbH.

operational constraints concerning in particular time windows, route durations, and user ride times. Consideration of different objectives and constraints leads to diverse variants of the DARP as studied in the literature, see e.g., [8,11].

In this work, we consider specifically the *electric autonomous dial-a-ride problem* (E-ADARP) as first introduced by Bongiovanni et al. [2]. The E-ADARP is a challenging but highly practically relevant extension to the DARP, where electric autonomous vehicles (EAVs) are employed. The adoption of autonomous vehicles without human drivers removes the restrictions on route durations and allows for continuous service. In addition to the ride-sharing, electric vehicles (EVs) are another step towards environment-friendlier mobility but their charging requirements have to be taken into account. Route schedules now generally need to include stops at charging stations to (partially) recharge the electric vehicles' batteries and constraints regarding the battery capacity and state of charge (SOC) must be fulfilled. Extending the already NP-hard standard DARP with these and further constraints as well as with the consideration of user inconvenience by including user excess ride times in the objective function further increases the problem's practical solving complexity substantially.

Bongiovanni et al. [2] proposed a mixed integer linear programming (MILP) based approach for the E-ADARP, which they successfully applied to instances with up to five vehicles and 50 customers. As this exact approach has substantial difficulties to scale to larger instances and therefore has only limited practical relevance, Su et al. [15] proposed a deterministic annealing (DA) metaheuristic for solving the E-ADARP. This approach applies an exact route evaluation scheme of linear time complexity based on a forward labeling algorithm. The authors introduce, as a central aspect, the concept of a battery-restricted fragment, which represents a subsequence of pickup and drop-off locations for which the minimum user excess ride time can be independently optimized. Large neighborhood search (LNS) [12] is a metaheuristic that is frequently applied with great success to diverse DARP variants. So did Bongiovanni et al. [1] for the E-ADARP. Moreover, Limmer [9] proposed a bilevel LNS (BI-LNS) variant in which the outer level employs multiple operators to schedule the charging of the EVs first and the inner level greedily inserts customer requests into the routes. This approach is highly scalable, solves instances with up to thousands of requests in reasonable time, and yields on large and very large benchmark instances results that are mostly superior to the earlier approaches.

In a previous work [3], we followed this line of research and developed an LNS for the E-ADARP that combines and significantly extends concepts from the mentioned earlier works. As an alternative to directly dealing with charging stops in the destroy and repair operators of the LNS, we proposed a novel route evaluation procedure that inserts charging stops as needed on-the-fly while also determining respective charging times. As a baseline approach, we formulated the charging station insertion subproblem as a MILP, but we also proposed a computationally more efficient heuristic. The performance of these approaches was studied on two sets of benchmark instances and the heuristic version yielded new state-of-the-art results for most instances.

In this work, we investigate the heuristic approach further and provide more information on our preprocessing of E-ADARP instances and the tuning of important LNS parameters. We consider additional configurations of the benchmark instances for evaluation and the results illustrate the general applicability of our approach as it finds (new) best solutions for almost all instances. The impact of different repair operators is analyzed, allowing for identification of the best performing one as well as of improvement opportunities.

2 Electric Autonomous Dial-A-Ride Problem

In the E-ADARP as originally defined in [2], n customer requests have to be served by n_K EAVs, given as set $K = \{1, \ldots, n_K\}$, within a planning horizon of T^{plan} time units. The problem is modeled on a complete directed graph $G = (V, A)$ where all considered geographic locations constitute the vertex set $V = N \cup O \cup F \cup S$ and the arc set A is defined as $A = \{(i, j) : i, j \in V, i \neq j\}$. Location subset $N = P \cup D$ consists of all customer pickup locations $P = \{1, \ldots, n\}$ and their corresponding drop-off locations $D = \{n + 1, \ldots, 2n\}$ such that the i-th request is given by a pair $(i, i + n)$. Each request $i \in P$ is associated with a maximum user ride time u_i that has to be respected. The route of each vehicle $k \in K$ starts at an origin depot $o_k \in O$ and ends at a destination depot $f_k \in F$, with $|O| = |F| = n_K$. Available charging station (CS) locations are denoted by set S, and each station $s \in S$ is assigned a charging rate α_s, specifying the amount of energy charged per time unit.

Earliest and latest possible service start times w_i^{start} and w_i^{end} are given for each vertex $i \in V$ and together they constitute the corresponding time window $[w_i^{\text{start}}, w_i^{\text{end}}]$. The service duration d_i is nonnegative at customer locations $i \in N$ and zero at all other locations. The change in load l_i of a vehicle serving a location i is positive at pickups, negative at drop-offs, and zero otherwise. The maximum load capacity of a vehicle $k \in K$ is denoted by C_k, and Q is the homogeneous battery capacity of each vehicle. A vehicle k starts from its origin depot with an initial battery level $B_{k,1}$ and has to arrive at its destination depot with a minimum battery level γQ, where $\gamma \in [0, 1]$ is the minimum battery level ratio. Traveling an arc $(i, j) \in A$ consumes $\beta_{i,j}$ battery and takes $t_{i,j}$ time, such that the triangle equality holds for both.

The *route* of a vehicle is a path from its origin to its destination depot, optionally going through pickup, drop-off, and charging station locations, and a corresponding *schedule* assigns a service start time t_i^{serv} to each node i of the route. The set of n_K vehicle routes together with their schedules constitute an E-ADARP solution, which is called *feasible* if it additionally satisfies the following constraints. The maximum capacity C_k of an EAV cannot be exceed at any time and its battery level has to be within $[0, Q]$ at all times. Besides, each charging station can only be visited by empty EAVs and at most once over all routes.

The goal is to find a feasible solution that minimizes the following weighted sum objective considering the total travel time and total user excess ride time

over all routes and requests:

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

with weight factors w^{routing} and w^{excess} and binary decision variables $x_{i,j}^k$ denoting sequential visits of vehicle k to locations i and j. The difference between the actual ride time of a user $i \in P$ and their minimum travel time $t_{i,i+n}$ is called the *user excess ride time*; i.e., $t_i^{\text{excess}} = t_{i+n}^{\text{serv}} - t_i^{\text{serv}} - d_i - t_{i,i+n}$.

3 Preprocessing

To reduce the complexity of E-ADARP instances, we employ two techniques proposed by Dumas et al. [6] for the pickup and delivery problem (PDP) with time windows and by Cordeau [4] for the DARP: time window tightening and arc elimination. Time window tightening is used to possibly reduce the span of given time windows. We use the extended rules for the E-ADARP given in [15] with the difference that we also take into account potentially already given time windows for charging stations as well as depots when considering these types of locations. We would also like to point out an inaccuracy in one of the rules in [15] for depots $i \in O \cup F$: $w_i^{\text{end}} := \min(w_i^{\text{end}}, \max(w_j^{\text{end}} + \boldsymbol{d_i} + t_{j,i}))\ \forall j \in D$ should be $w_i^{\text{end}} := \min(w_i^{\text{end}}, \max(w_j^{\text{end}} + \boldsymbol{d_j} + t_{j,i}))\ \forall j \in D$.

Arc elimination removes arcs that cannot be part of a feasible solution due to time window, ride time, and other constraints. As a base, we use the rules from [7] for the PDP with time windows and electric vehicles and enhance them by accounting for the existence of multiple origin and destination depots. We also employ a new rule based on vehicle loads that eliminates an arc (i, j) between two pickup nodes i and j if the sum of the demands of requests i and j exceeds the maximum vehicle capacity: $l_i + l_j > \max_{k \in K}(C_k)$. Additional path-based elimination rules from [4,6] enable further arc removals as well as the disclosure of incompatible request pairs, which cannot be part of the same route. This information allows for immediate rejection of certain insertion positions of a request into a route during the LNS repair process and thus a speedup thereof.

4 Large Neighborhood Search

In this section, we briefly describe our proposed large neighborhood search based solution approach for solving the E-ADARP but refer to [3] for more details. An initial solution is obtained by first creating for each vehicle a route from its origin to its destination depot and then feasibly inserting as many requests as possible with the time window order based repair operator described below. The resulting solution is feasible except that some requests may still be unserved. In each iteration, random removal [14] is used to pick and delete κ served requests from the routes, which are then tried to be reinserted by one of three repair operators. (a) A classic greedy heuristic [14] repeatedly evaluates every possible combination

of inserting the pickup and drop-off locations of every unserved request into each route and selects the option with a minimum cost increase each time. (b) A less computationally intensive variant is used in the second operator, where the unserved requests $i \in P$ are sorted and inserted in their cheapest feasible position in non-decreasing order of w_i^{start}. (c) The third operator works in the same way except that a random order is applied to potentially escape local optima. During the repair and when evaluating candidate routes, their costs are increased by a random noise term to promote diversification as also seen in [9,14].

On-The-Fly (OTF) Charging Station Insertion. The employed LNS operators do not deal with the insertion of charging stations (CSs) into the (candidate) routes but instead, this is done on-the-fly as needed during the route evaluation, which also handles the scheduling including the computation of charging times. We refer to [3] for a detailed description including the formulation of the corresponding charging station insertion and evaluation subproblem as a mixed integer linear program and only outline the time-efficient heuristic solving approach in the following. First, all charging stops are removed from the route at hand, before iterating twice over all its stops. This is done once in a forward- and once in a backward-pass during which all necessary information is computed to determine potential time window and battery constraint violations as well as the range of possible CS insertion positions. If a charging stop is needed, all combinations of available CSs and insertion positions are tested for feasibility. If there is at least one option satisfying the battery constraints, the one with the shortest incurred detour is selected, otherwise we pick the one where the vehicle can charge the most energy. After insertion of the respective charging stop into the route, all affected data is updated and the procedure is repeated, thus, further CSs are possibly inserted, until the (in-)feasibility of the route is determined. For details, we refer to [3].

5 Experimental Analysis

The proposed approach was implemented in Julia 1.10.0 with Gurobi 10.0.3[1] in single-threaded mode as MILP solver. All tests were run on single cores of 2.4 GHz Intel Xeon E5-2640 v4 processors with a memory limit of 20 GB and a time limit of 300 s. We employ two sets of DARP benchmark instances by Cordeau[2] [4] and Ropke[3] [13], which are enhanced with E-ADARP features and follow the naming scheme "an_K–n", where n_K is the number of vehicles and n the number of requests. The weights in the objective function (1) are set to $w^{\text{routing}} = 0.75$ and $w^{\text{excess}} = 0.25$ as in [2,9,15]. We consider three different minimum final battery level ratios $\gamma \in \{0.1, 0.4, 0.7\}$ and a restriction to one visit per CS, $n_{\text{s}} = 1$, as stated in the original problem definition. For each instance, our LNS is run 30 times for each configuration.

[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

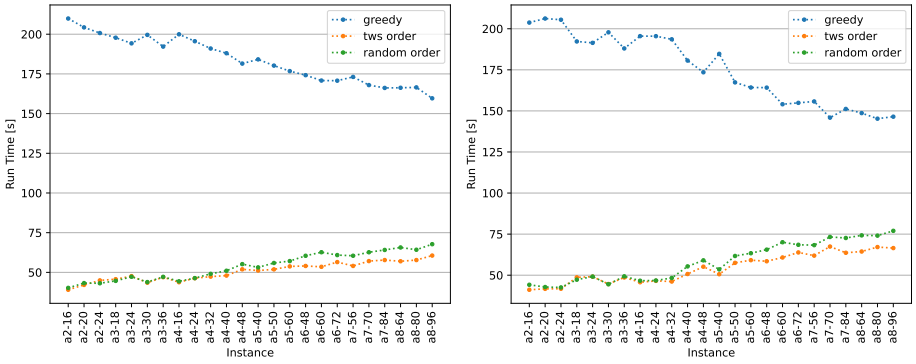
Table 1: Results on Cordeau and Ropke instances with different values for γ and with limited CS visits ($n_s = 1$).

Instance	$\gamma = 0.1$				$\gamma = 0.4$				$\gamma = 0.7$			
	BKS	OTF Heuristic			BKS	OTF Heuristic			BKS	OTF Heuristic		
		Obj min	Obj mean	Obj sd		Obj min	Obj mean	Obj sd		Obj min	Obj mean	Obj sd
Cordeau												
a2-16	237.38	237.38	**237.38**	0.00	237.38	237.38	**237.38**	0.00	240.66	240.66	**240.66**	0.00
a2-20	279.08	279.08	**279.08**	0.00	280.70	280.70	**280.70**	0.00	293.27	293.27	**293.27**	0.00
a2-24	346.21	346.21	**346.21**	0.00	347.04	349.20	349.20	0.00	353.18	353.18	**353.18**	0.00
a3-18	236.82	236.81*	**236.81**	0.00	236.82	236.81*	**236.81**	0.00	240.58	240.58	**240.58**	0.00
a3-24	274.80	274.80	**274.80**	0.00	274.80	274.80	**274.80**	0.00	275.97	275.97	**275.97**	0.00
a3-30	413.27	413.27	**413.27**	0.00	413.34	413.37	413.37	0.00	424.93	424.93	426.12	1.59
a3-36	481.17	481.17	482.18	1.37	483.06	483.06	485.92	2.78	494.04	494.04	497.18	2.61
a4-16	222.49	222.49	**222.49**	0.00	222.49	222.49	**222.49**	0.00	223.13	223.13	**223.13**	0.00
a4-24	310.84	310.84	**310.84**	0.00	311.03	311.03	**311.03**	0.00	316.65	316.65	**316.65**	0.00
a4-32	393.96	393.95*	**393.95**	0.00	394.26	394.26	**394.26**	0.00	397.87	397.87	**397.87**	0.00
a4-40	453.84	453.84	454.46	1.95	453.84	453.84	454.84	1.87	467.72	467.72	474.47	6.23
a4-48	554.54	554.54	555.38	0.73	554.60	554.60	556.98	1.49	575.35	575.62	579.63	2.40
a5-40	414.51	414.50*	414.99	0.91	414.51	414.50*	415.12	1.05	418.75	418.75	421.16	3.25
a5-50	559.17	559.17	562.18	2.40	560.41	559.51*	564.41	3.44	589.61	589.61	596.09	3.87
Ropke												
a5-60	691.83	683.87*	687.42	2.50	688.16	685.51*	690.63	2.94	NA	NA	NA	NA
a6-48	506.72	506.45*	506.77	0.21	506.85	506.45*	506.84	0.30	517.12	517.12	521.62	2.95
a6-60	692.00	690.29*	693.54	2.14	692.69	690.29*	693.16	2.31	714.16	714.16	731.45	10.02
a6-72	777.44	762.16*	770.69	3.77	771.97	765.64*	776.00	4.47	NA	NA	NA	NA
a7-56	613.10	612.53*	614.70	2.30	613.66	612.78*	615.27	2.51	636.56	636.56	649.37	10.43
a7-70	760.90	756.27*	761.16	3.23	761.62	756.46*	760.21	2.03	816.64	816.64	840.59	16.29
a7-84	889.38	874.57*	883.45	4.79	886.19	878.99*	890.18	7.23	NA	NA	NA	NA
a8-64	641.99	632.21*	637.93	3.89	637.95	632.95*	639.02	3.05	639.06	639.06	651.33	6.63
a8-80	803.52	793.64*	802.86	4.42	793.17	794.04	800.85	5.13	837.79	837.79	862.75	15.14
a8-96	1053.11	1032.76*	1041.59	4.74	1048.72	1036.22*	1047.47	5.62	NA	NA	NA	NA

LNS Parameter Tuning. Identifying suitable and robust parameter settings is crucial for the performance of the proposed LNS-based approach, so we use a selection of ten representative instances from the Cordeau and Ropke sets and the open source tool SMAC3[4] [10] for automated tuning. Here, we consider the degree of destruction κ that specifies the number of elements selected in the destroy operator and the noise rate η, which controls how much noise is used for the route evaluation in the repair operators as the noise is randomly sampled from $[-\eta t^{\max}, \eta t^{\max}]$ with $t^{\max} = \max_{i,j \in V} t_{i,j}$. Both parameters are simultaneously tuned by running 10000 trials of our LNS algorithm with possible value ranges of $[3, 96]$ for κ and $[0.00, 0.05]$ for η, leading to a final configuration of $\kappa = 15$ and $\eta = 0.014$.

Results. An overview of the performance of our LNS with the on-the-fly (OTF) insertion heuristic regarding the different minimum battery level ratios is given in Table 1. Columns "BKS" show the best known solution values considering results reported in [2,3,9,15]. For our approach, we list minimum and average objective values as well as the standard deviations (sd) over all feasible runs.

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

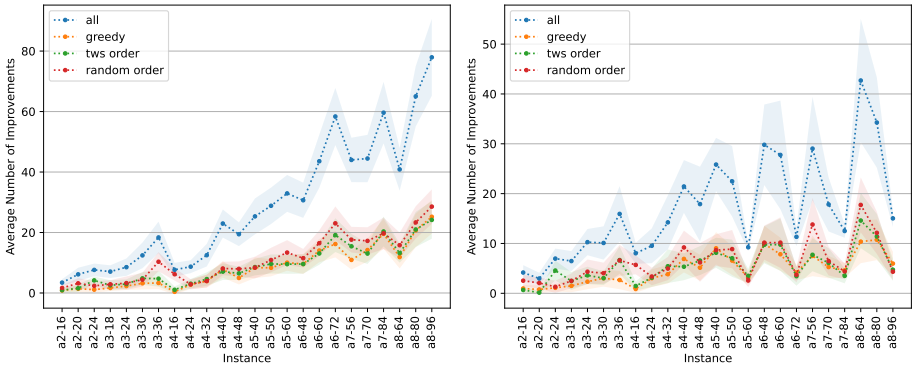


(a) $\gamma = 0.1$. (b) $\gamma = 0.7$.

Fig. 1: Average run time in seconds of the different LNS repair operators over 30 runs for each instance with limited CS visits ($n_s = 1$).

Results with an asterisk (*) indicate newly found best objective values and bold numbers highlight cases, where the mean objective value coincides with the (new) best known objective value, i.e., where the OTF heuristic yielded the best solution every time. The latter is the case for at least half of the Cordeau instances irrespective of the employed γ-value, although an instance generally becomes harder to solve with increasing values for γ resulting in higher objective values. This illustrates the applicability of our approach over different instance configurations. The results further show that the OTF heuristic is able to find the best solutions for almost all instances and configurations and how it excels especially on the Ropke instances. The impact of the different minimum battery level ratios on the objective values and the solvability is also higher on these larger instances. For the largest tested value of $\gamma = 0.7$, none of the considered approaches, including ours, could solve four of the instances in any trial, as denoted by values "NA", whereas our OTF heuristic solved all other instances in every trial. This strongly suggests that the respective four instances are infeasible.

Besides, we also investigate the performance of the LNS and the contribution of the different repair operators in terms of the mean run time as well as the average number of improvements over all runs for an instance, where an improvement denotes a new best solution during the search process. As the repair method is selected uniformly at random in each iteration, all operators are applied about equally often. The results for the smallest and largest tested γ-values are illustrated in Figures 1 and 2 for the run time and the improvements respectively, where "tws order" stands for the time window start based operator. From Figures 1a and 1b, we conclude that the value of γ does not substantially influence the time spent in each operator. As expected, the greedy operator is consistently the computationally most expensive one whereas the other two have similarly low run times. Figure 2 shows that all operators exhibit a similar performance regarding the mean number of improvements with the random



(a) $\gamma = 0.1$. (b) $\gamma = 0.7$.

Fig. 2: Average number of improvements of the different LNS repair operators over 30 runs for each instance with limited CS visits ($n_s = 1$).

Table 2: Performance of LNS repair operators over all instances and runs with a feasible initial solution for different values for γ.

Operator	$\gamma = 0.1$						$\gamma = 0.7$					
	Runs	Applications	Improve-ments	Avg. obj gain	Avg. obj gain / impr	Time [s]	Runs	Applications	Improve-ments	Avg. obj gain	Avg. obj gain / impr	Time [s]
greedy	659	216576.59	8.90	-0.000169	-3.21	182.72	172	269917.33	2.87	-0.000047	-3.61	193.76
tws order	659	216592.04	9.58	-0.000166	-2.75	50.98	172	269907.62	3.36	-0.000059	-3.86	46.83
random order	659	216554.78	11.27	-0.000178	-2.57	54.52	172	269891.45	4.63	-0.000060	-2.96	48.05

order based one being slightly more successful in most cases. Thus, investing more time in the greedy operator does not seem to pay off. We also observe a difference in the pattern of the overall performance for the different instance configurations. In both subfigures, the mean number of improvements is lower for smaller instances, which is due to these instance generally being solved in less iterations. For larger instances, there is an increase in improvements with the number of vehicles and requests respectively in Figure 2a, which indicates consistent improvements during the LNS. In Figure 2b with the larger γ however, the number of improvements drops regarding a steady number of vehicles while increasing the amount of requests. As pairing this γ-configuration with more requests results in harder instances and considering that the overall number of LNS iterations does not decline compared to the easier configuration, the observations here suggest that our operators might get stuck in local optima earlier, leaving room for further improvement in this regard.

To gain more insights, we employ further performance metrics for the repair operators such as the average objective gains per application and per improvement. These and the previous metrics are reported in Table 2 over all instances and runs where the construction heuristic yields a feasible initial solution. We consider this restriction to avoid bias in our data towards operators finding the first feasible

solution, which represents a disproportional large objective gain in our approach. The results show that the mean objective gain per application is in general quite low, which is due to a high number of performed LNS iterations compared to relatively few achieved improvements. Regarding this metric, the random order based operator performs the best, which is in contrast to its performance in terms of average gain per improvement where it achieves the lowest value of all operators. For the greedy operator, we observe again that it finds on average less improvements but when it does, the gain is often higher and especially so for $\gamma = 0.1$. Considering the larger γ-value, the time window order based operator yields the highest mean gain per improvement despite consuming the least time.

6 Conclusions and Future Work

In this work, we considered the electric autonomous dial-a-ride problem and further investigated our LNS-based solving approach with an on-the-fly charging station insertion heuristic. We detailed our preprocessing procedure for reducing E-ADARP instance sizes by adapting and extending reduction rules from the literature. For the performance evaluation, we considered various configurations of minimum battery level ratios, which influence the hardness and solvability of instances. The new results confirm our previous findings on the general success of the OTF heuristic and especially so on larger instances with up to 96 requests and eight EAVs. We observed that the repair operator which considers requests for insertion in a random order performs best in terms of the average number of improvements but yields the lowest objective gain per improvement. Regarding the latter metric, the greedy and time window order based operators achieve the best results depending on the employed γ-value.

In the future, we want to further improve the scalability of our approach to tackle huge instances with possibly thousands of requests, e.g., by sparsifying the underlying graph, restricting the neighborhoods, or employing more advanced methods for selecting the destroy sets. Utilization of machine learning techniques could be a promising research direction for this.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bongiovanni, C., Kaspi, M., Cordeau, J.F., Geroliminis, N.: A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations. Transportation Research Part E: Logistics and Transportation Review **165**, 102835 (2022). https://doi.org/10.1016/j.tre.2022.102835
2. Bongiovanni, C., Kaspi, M., Geroliminis, N.: The electric autonomous dial-a-ride problem. Transportation Research Part B: Methodological **122**, 436–456 (2019). https://doi.org/10.1016/j.trb.2019.03.004

3. Bresich, M., Raidl, G.R., Limmer, S.: Letting a large neighborhood search for an electric dial-a-ride problem fly: On-the-fly charging station insertion. Genetic and Evolutionary Computation Conference (GECCO '24) (2024). https://doi.org/10.1145/3638529.3654057
4. Cordeau, J.F.: A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. Operations Research **54**(3), 573–586 (2006). https://doi.org/10.1287/opre.1060.0283
5. Cordeau, J.F., Laporte, G.: A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological **37**(6), 579–594 (2003). https://doi.org/10.1016/S0191-2615(02)00045-0
6. Dumas, Y., Desrosiers, J., Soumis, F.: The pickup and delivery problem with time windows. European Journal of Operational Research **54**(1), 7–22 (1991). https://doi.org/10.1016/0377-2217(91)90319-Q
7. Goeke, D.: Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. European Journal of Operational Research **278**(3), 821–836 (2019). https://doi.org/10.1016/j.ejor.2019.05.010
8. Ho, S.C., Szeto, W.Y., Kuo, Y.H., Leung, J.M.Y., Petering, M., Tou, T.W.H.: A survey of dial-a-ride problems: Literature review and recent developments. Transportation Research Part B: Methodological **111**, 395–421 (2018). https://doi.org/10.1016/j.trb.2018.02.001
9. Limmer, S.: Bilevel large neighborhood search for the electric autonomous dial-a-ride problem. Transportation Research Interdisciplinary Perspectives **21**, 100876 (2023). https://doi.org/10.1016/j.trip.2023.100876
10. Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., Hutter, F.: SMAC3: A versatile bayesian optimization package for hyperparameter optimization. Journal of Machine Learning Research **23**(54), 1–9 (2022), http://jmlr.org/papers/v23/21-0888.html
11. Molenbruch, Y., Braekers, K., Caris, A.: Typology and literature review for dial-a-ride problems. Annals of Operations Research **259**(1-2), 295–325 (2017). https://doi.org/10.1007/s10479-017-2525-0
12. Pisinger, D., Ropke, S.: Large neighborhood search. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, pp. 399–419. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-1-4419-1665-5_13
13. Ropke, S., Cordeau, J.F., Laporte, G.: Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks **49**(4), 258–272 (2007). https://doi.org/10.1002/net.20177
14. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science **40**(4), 455–472 (2006). https://doi.org/10.1287/trsc.1050.0135
15. Su, Y., Dupin, N., Puchinger, J.: A deterministic annealing local search for the electric autonomous dial-a-ride problem. European Journal of Operational Research **309**(3), 1091–1111 (2023). https://doi.org/10.1016/j.ejor.2023.02.012