


# Search Space Reduction Through Machine Learning for the Electric Autonomous Dial-A-Ride Problem

Maria Bresich<sup>1</sup><sup>[0009-0000-8291-6765]</sup>\*, Günther R. Raidl<sup>1</sup><sup>[0000-0002-3293-177X]</sup>,  
Caspian Coleman<sup>1</sup><sup>[0009-0008-7605-8010]</sup>, Pascal Welke<sup>2,1</sup><sup>[0000-0002-2123-3781]</sup>,  
and Steffen Limmer<sup>3</sup><sup>[0000-0003-2385-7886]</sup>

<sup>1</sup> TU Wien, Vienna, Austria

<sup>2</sup> Lancaster University Leipzig, Leipzig, Germany

<sup>3</sup> Honda Research Institute Europe GmbH, Offenbach/Main, Germany  
`{mbresich|raidl}@ac.tuwien.ac.at, caspiancoleman@outlook.com,`  
`p.welke@lancaster.ac.uk, steffen.limmer@honda-ri.de`

**Abstract.** The complexity of combinatorial optimization problems typically leads to a steep performance decrease of solving approaches with increasing problem size. We explore the usage of machine learning (ML) to substantially reduce the search space size of large problem instances by predicting and removing unpromising parts in order to accelerate the subsequent solving process. More specifically, we explore graph sparsening techniques for the electric autonomous dial-a-ride problem (E-ADARP), where self-driving electric vehicles are used to provide an efficient and sustainable ride-sharing service by serving customer transportation requests between pickup and drop-off locations within specified time windows. Approaches utilizing support vector machines as well as gradient boosted trees are compared to and also combined with a common  $k$ -nearest neighbor heuristic. Our goal is to boost the performance of a state-of-the-art large neighborhood search for the E-ADARP to make it well applicable to instances with up to 5200 requests and 260 vehicles. Our ML models are trained on representative instances and close-to-optimal solutions obtained from excessively long runs in a weakly supervised fashion. We uncover challenges, a fundamental limitation, and benefits of this approach and are able to achieve the intended scalability.

**Keywords:** Dial-A-Ride Problem · Electric Autonomous Vehicles · Problem Space Pruning · Graph Reduction · Machine Learning

## 1 Introduction

Electric autonomous vehicles (EAVs) are a promising answer for current challenges in transport such as increasing traffic volumes and a growing market for on-demand transportation. Utilizing a fleet of EAVs to provide a sustainable

---

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

ride-sharing service could keep travel individual and flexible while reducing the high cost of private car ownership, using available capacities more efficiently than classical taxi services, and reducing the environmental impact. The corresponding routing problem is called the electric autonomous dial-a-ride problem (E-ADARP) [7], where the aim is to find minimum cost routes that serve all customer requests. The E-ADARP is an NP-hard combinatorial optimization problem, and its complexity makes calculating optimal solutions in general practically infeasible when considering more than five vehicles and fifty customer requests [7]. Most existing heuristic approaches in the literature focus only on up to twice as many but to have a considerable impact on urban traffic, scalable approaches are essential to be able to handle much larger fleets and customer volumes efficiently.

In this work, our goal is to improve the scalability of a sophisticated large neighborhood search (LNS) for the E-ADARP [8] that is leading on instances with up to 96 requests to scenarios that are magnitudes larger with up to 5200 orders [21]. To deal with the bottleneck introduced by the huge number of arcs in the underlying instance graph, we substantially prune the search space by heuristically sparsifying the graph, keeping only promising arcs that are likely to appear in close-to-optimal solutions. This is a commonly applied strategy to improve the efficiency of the subsequent optimization, e.g., simple  $k$ -nearest neighbor ( $k$ -NN) sparsification [12] is a standard strategy on the traveling salesman problem (TSP). Lately, tackling such pruning decisions by machine learning (ML) techniques has gained increasing attention in the literature [14, 28, 30]. In contrast to the TSP and structurally simpler vehicle routing problems (VRPs) considered in existing works, the highly constrained E-ADARP constitutes a much more challenging and advanced problem formulation where it is not straightforward to define under which conditions an arc should be considered promising. We follow and extend the lines of previous research and explore sparsification approaches of individually applying as well as combining  $k$ -NN and ML techniques in the context of the E-ADARP. The main idea is to train an ML model in a weakly supervised offline fashion on high-quality solutions for many representative instances and to use the model to decide which arcs to keep or discard. We encounter challenges and questions related, for example, to suitable features and the interplay of sparsification and the inner workings of an LNS. Training ML models in such a context works in general reasonably well but when applied within the LNS framework, some simpler models using fewer specific features perform notably better, despite their clearly worse prediction performance. Thus, a more fundamental limitation of the general approach exists. Further analysis of these circumstances reveals interesting insights and explanations. Ultimately, we are able to boost the performance of the employed LNS with a simpler learned model and beat the LNS by Limmer [21] that was leading for such large benchmark instances so far.

The remainder is organized as follows. The next section introduces the E-ADARP formally. Section 3 reviews related work. Section 4 presents the proposed approaches for search space pruning, including design motivations, used features, and ML models. Experimental results are shown and discussed in Section 5. Finally, Section 6 concludes the paper and outlines promising future work.

## 2 Electric Autonomous Dial-a-Ride Problem

Following Bongiovanni et al. [7], the E-ADARP is defined on a weighted directed graph  $G = (V, A)$ , with vertex set  $V$  representing all relevant locations and arc set  $A = \{(i, j) : i, j \in V, i \neq j\}$  the direct connections in between. Each arc  $(i, j) \in A$  is associated with a corresponding travel time  $t_{i,j}$  and an energy (battery) consumption  $\beta_{i,j}$ . We are given a fleet of  $n_H$  homogeneous vehicles, denoted by  $H = \{1, \dots, n_H\}$ , which has to provide service for  $n$  customer requests. The pickup locations of these requests are  $P = \{1, \dots, n\} \subset V$  and the drop-off locations  $D = \{n+1, \dots, 2n\}$ , respectively, and each request  $i = 1, \dots, n$  has a time window  $[w_i^{\text{start}}, w_i^{\text{end}}]$  for the service start time as well as a maximum ride time  $u_i \geq t_{i,i+n}$ . Additional vertex sets  $O \subset V$  and  $F \subset V$  represent the vehicles' origin and destination depots, respectively, and  $S \subset V$  represents available charging stations (CSs). A charging rate  $\alpha_s$  for each charging station  $s \in S$  defines the amount of energy that can be charged per time unit. Moreover, each location  $i \in V$  has assigned a service duration  $d_i$  with  $d_i \geq 0$  for pickup and drop-off locations  $i \in P \cup D$  and  $d_i = 0$  otherwise. Vehicles  $h \in H$  have a uniform maximum battery capacity  $Q$  and individual initial battery levels  $B_{h,1}$  at the start of the planning horizon, and they need to meet a minimum battery level  $\gamma Q$  at the end of the planning horizon with  $\gamma \in [0, 1]$  denoting the minimum end state-of-charge (SoC). Vehicles have a maximum load capacity  $C$ , and we assume here that each request requires one unit load to be transported, as this is the case in the common E-ADARP benchmark instances we will consider.

A *route* of a vehicle is a feasible sequence of locations  $i \in V$  starting at its origin depot and ending at its destination depot. An E-ADARP solution consists of  $n_H$  vehicle routes together with a service start time  $t_i^{\text{serv}}$  for each location  $i$  in the routes. For visited charging stations, the durations of charging have to be optimized as well. For a solution to be feasible, each request must be served exactly once; i.e., for each request  $i = 1, \dots, n$ , there is one route that visits pickup location  $i$  before drop-off location  $i+n$ . Additionally, the maximum user ride times and maximum load and battery capacities must never be exceeded, the time windows not be violated, and the EAVs must not run out of charge at any time. Charging stations can be visited an unlimited number of times but only by EAVs with no passenger on board.

The E-ADARP aims to find a feasible solution that minimizes a linear combination of the total travel time over all routes and the total user excess ride time over all requests:

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

where  $w^{\text{routing}}$  and  $w^{\text{excess}}$  are weighting factors for the two components. Binary decision variables  $x_{i,j}^h$  indicate whether vehicle  $h \in H$  visits locations  $i$  and  $j$  in direct sequence, for  $(i, j) \in A$ . The user excess ride time  $t_i^{\text{excess}}$  for request  $i$  is the difference between the actual ride time and the time  $t_{i,i+n}$  it takes to travel directly from the pickup to the drop-off location of  $i$ .

### 3 Related Work

The E-ADARP was initially introduced by Bongiovanni et al. [7], where a three- and a two-indexed mixed-integer linear programming (MILP) formulation were presented. In later works, the same authors designed a scheduling algorithm to minimize the users' ride time [5] and tackled a dynamic variant by iteratively solving static E-ADARP subinstances by employing a two-phase metaheuristic including a machine learning-based large neighborhood search [6].

Su et al. [24] proposed a deterministic annealing metaheuristic for the E-ADARP and introduced the notion of battery-restricted fragments for representing E-ADARP routes, which are used to minimize the user excess ride time. In [23], the same authors further build upon this concept and propose a path-based MILP formulation that is solved with Branch-and-Price; a labeling algorithm is used for column generation.

Limmer [21] proposed a bilevel LNS (BI-LNS) that separates the optimization of EAV charging and request handling by first scheduling charging stops in the outer level before dealing with the optimization of requests in the inner level. This work was also the first to introduce very large-scale problem instances with thousands of requests on which the scalability of BI-LNS is demonstrated.

Lastly, Bresich et al. [8] presented two advanced variants of an LNS, one of which employs a novel route evaluation procedure for inserting charging stops on-the-fly as needed in a close-to-optimal fashion. In this way, the used destroy and repair operators of the LNS can be kept relatively simple as they do not have to directly consider charging stops: The destroy operator removes a certain number of randomly selected requests from the current solution and three variants of repair operators insert them back into the routes by following different greedy strategies. This metaheuristic also utilizes the fragment-based route representation from [24], and for the insertion of charging stops, only positions in between fragments need to be considered. Available positions and charging stations are iteratively tested for feasibility and selected according to their potential amount of energy charged and incurred detour length. This on-the-fly LNS (OTF-LNS) approach currently is the state-of-the-art for most of the common E-ADARP benchmark instance sets. Only on the very large instances from Limmer [21], OTF-LNS falls behind BI-LNS as OTF-LNS actually cannot be meaningfully applied anymore in a direct fashion due to its much too high runtime requirements. The better scalability of BI-LNS can be explained by its simplicity and therefore higher time-efficiency. In the current work, we build upon OTF-LNS and aim at improving its scalability to the very large instances from [21] by applying search space pruning techniques.

Sparsification of the underlying graph of transportation problems is a common approach to reduce the search space and improve the scalability of solution approaches. Exact methods only remove elements that can provably never occur in feasible solutions. Examples are arc elimination rules [9, 11] that are commonly used for diverse VRPs. All above mentioned works for the E-ADARP including in particular [8] apply variants of these deterministic pruning rules and achieve significant reductions of instance graphs. Still, the remaining graphs are often rather dense, and much potential is left for heuristic pruning techniques. A

well-known heuristic is  $k$ -nearest neighbor sparsification [12], which keeps only the  $k$  shortest incident edges for each node and discards the rest. It has been successfully employed in heuristic solving approaches for routing problems such as the TSP and basic VRP variants [2, 3, 17, 22]. Recent applications extend to end-to-end deep learning-based solving approaches for the TSP and VRP [1, 18]. In our work, we consider a variant of the  $k$ -NN sparsification as seen in Bertsimas et al. [4], which takes into account that the underlying graph is directed.

*Learning-to-prune*, i.e., using ML techniques to reduce the instance size and hence the search space of heuristics, has been applied to diverse combinatorial optimization problems. Fitzpatrick et al. [13] utilized advanced features derived from the linear programming relaxation, cutting planes, and more on the TSP. Sparsification rates of over 85% barely impacted solution quality and often even preserved optimality, although not in a guaranteed way. Sun et al. [25] showed that such an approach may generalize well to out-of-distribution instances.

Beyond the TSP, Lauri et al. [20] showed similar success for the maximum clique enumeration problem, removing up to 99% of nodes and demonstrating manifold speed-ups of subsequent solving approaches with only moderate loss in solution quality. Lauri and Dutta [19] introduced a multi-stage sparsification concept by training different classifiers for different stages of the pruning process. For the maximum weight clique problem, Sun et al. [26] investigated statistical measures and ML for problem reduction where they use ranking and correlation scores and features derived from the graph structure for training. ML-based methodologies for search space pruning and heuristic guidance show success in both prediction accuracy and run time reduction for various NP-hard problems such as the minimum vertex cover [28], maximum independent set [28], and uncapacitated facility location [30] problems.

Fitzpatrick et al. [14] present the only ML-based pruning approach in the context of VRPs known to the authors. They considered specifically an electric VRP and trained a linear support vector machine, a random forest classifier, and a logistic regression model to predict which arcs of the underlying graph are likely to appear in near-optimal solutions. For training, they applied weak supervision as a way to deal with more complex routing problems by replacing the need for optimal solutions with just near-optimal solutions. In our work, we follow this line of research and extend it to the structurally more challenging E-ADARP. As the instances we are focusing on are magnitudes larger, a fast feature computation and prediction are crucial, and we therefore rely on a comparatively small subset of simple features derived directly from the instance and also investigate different types of ML models and combinations with  $k$ -NN sparsening.

## 4 Search Space Pruning for the E-ADARP

We aim at pruning E-ADARP instances by sparsification of the underlying graph  $G = (V, A)$ , specifically the removal of a substantial portion of arcs that are assumed to be unlikely to appear in close-to-optimal solutions. We rely on OTF-LNS [8] as underlying optimization approach for the E-ADARP, and

our primary motivation is to improve its scalability to very large-scale problem instances such as those from Limmer [21] with up to 5200 requests and 260 EVs. The computational bottleneck of OTF-LNS is the way how removed requests are reinserted into the current, partially destroyed solution: The pickup location of the request is tried to be inserted at every position of each route, followed by trying to insert the drop-off location of the request at every feasible position after the pickup in the same route. Assuming a maximum route length of  $n_R$  stops, this in essence boils down to  $O(n_H \cdot n_R)$  candidate routes to be evaluated for each request insertion. The route evaluation with the OTF charging station insertion is, in principle, highly efficient but still requires  $O(n_R)$  time, implying a total time for one LNS iteration, i.e., the removal of a small constant number of requests followed by their re-insertion, of  $O(n_H \cdot n_R^2)$ . By sparsening the graph  $G$  (in addition to the deterministic problem reduction performed by Bresich et al. [8]) and only further pursuing insertion options where the corresponding arcs for the insertion are still present, the number of candidate routes that undergo the OTF charging stop insertion and evaluation may be significantly reduced to the most promising ones, and therefore the LNS is substantially sped up.

It is important to distinguish between different types of arcs connecting different vertex types: depot arcs and CS arcs, which are incident to a depot or charging station respectively, and PD arcs, for which both end points are either pickup or drop-off locations. According to the structure of E-ADARPs, PD arcs constitute the majority of arcs with  $\Theta(n^2)$  many, whereas there are only  $(mn + |S|n)$  depot and CS arcs, with  $m$  and  $|S|$  being comparably small. We expect these types to have different properties regarding their probabilities to appear in good solutions. For simplicity, we restrict here the sparsification to the dominant PD arcs with one exception: The direct PD connections of the requests, i.e., arcs  $(i, i + n)$  for  $i \in P$ , are also always preserved to aid the existence of feasible solutions even in case of strong pruning.

As a baseline, we employ the common  $k$ -NN sparsification [12] with the travel time  $t_{i,j}$  between nodes  $i, j \in V$  as nearness criterion. The standard approach of keeping the  $k$  nearest neighbors is typically applied on undirected graphs or restricted to outgoing arcs on directed graphs. Considering the inner workings of the LNS, our aim is to maintain a minimum in- and outdegree for each node by conserving the  $k$  best in- as well as outgoing arcs in order to enable sufficient exploration opportunities during the search. This still results in a reduction with the number of remaining PD arcs being in  $O(kn)$ . An obvious weakness of this approach is that it only exploits travel times and does not care about other important properties of the complex E-ADARP structure, such as the time windows of requests. In the following, we therefore consider more sophisticated ML-based approaches that consider a variety of features derived from the instance structure and are trained in a weakly supervised manner.

Our fundamental ML-based approach as outlined in Algorithm 1 calculates a feature vector for every PD arc  $(i, j) \in A$  that remains after the deterministic problem reduction and applies a pre-trained model in order to decide whether to keep the arc or remove it, before applying the OTF-LNS. Translating this

**Algorithm 1:** Pseudo-code for ML-based pruning and solving approach.

---

**Input:** Instance  $\mathcal{I}$  with graph  $G = (V, A)$ , pre-trained model  $M$ , parameter  $k$

- 1  $\mathcal{I}', (V, A') \leftarrow \text{deterministic\_reduction}(\mathcal{I}, (V, A));$
- 2  $A^{\text{PD}} \leftarrow \text{get\_PD\_arcs}(A');$
- 3  $\text{features} \leftarrow \text{get\_arc\_features}(A^{\text{PD}});$
- 4  $A'^{\text{PD}} \leftarrow \text{prune}(\mathcal{I}', A^{\text{PD}}, \text{features}, M, k);$
- 5  $A'' \leftarrow A' \setminus A^{\text{PD}} \cup A'^{\text{PD}};$
- 6  $\text{OTF-LNS}(\mathcal{I}', (V, A''));$

---

into a binary classification task of distinguishing arcs into a promising and an unpromising class, referring to whether or not they are likely part of a high quality solution, seems natural. However, the number of arcs we should keep depends heavily on the allotted run time of the LNS and also the problem structure and size. In general, shorter run times will require stronger pruning to enable fast construction of an initial solution and more iterations, while longer run times allow for keeping more arcs and slower convergence. We thus consider that a threshold on the number of arcs to be kept shall be externally provided, and that the decision making has to be robust over a large range of thresholds. We therefore aim at approximating the probability of arcs to appear in good solutions, which is a regression instead of a classification task, and perform logistic regression to obtain a probabilistic heatmap over the arcs.

For the weakly supervised training, we consider a set of independent training instances that was created in the same randomized fashion as the E-ADARP instances from Limmer [21]. Due to the complexity of the E-ADARP, computing proven optimal solutions for such large instances is intractable, so instead we use reasonably good heuristic solutions identified by OTF-LNS [8]; cf. Section 5. As many close-to-optimal solutions may exist for an E-ADARP instance, not only a single solution is computed but rather multiple solutions are determined and all PD arcs appearing in at least one of the solutions are labeled positively, i.e., considered in-solution arcs. To account for the fact that the LNS does not operate on CS arcs directly as they are only introduced by the OTF heuristic during evaluation, we replace all in-solution arcs incident to a CS by corresponding PD arcs from the first preceding to the first succeeding pickup or drop-off node of the CS in the route.

*Features.* We consider quickly computable yet robust features for the arcs with the aim to achieve an effective pruning while keeping the introduced time overhead low. For the E-ADARP, the consideration of travel times  $t_{i,j}$  between nodes  $i, j \in V$  appears natural as they are a central part of the objective function, but it is unclear which other features would contribute to good predictions. We thus investigated a variety of features for PD arcs, including but not limited to the differences of time window start, end, and mid-points, the overlap and span of time windows, the  $\alpha$ -nearness [17], and the distance from the arc centroid to the nearest and farthest away depot or charging station. The associated battery

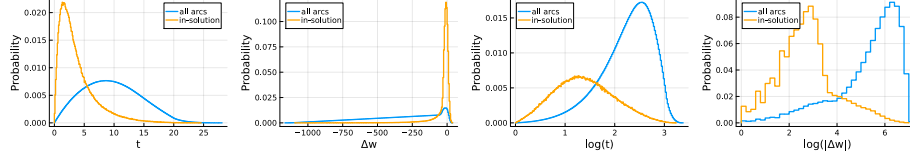


Fig. 1: Distributions of original features and after logarithmic transformations.

consumption and distance of an arc would be further natural features but in the considered benchmark instances, they are both proportional to the travel time, thus already implicitly covered. While multiple features showed potential during a detailed statistical analysis, we found in preliminary studies that the most suitable and important ones were the travel time  $t_{i,j}$  and the difference of time window mid-points  $\Delta w_{i,j} = \frac{w_i^{\text{start}} - w_j^{\text{start}} + w_i^{\text{end}} - w_j^{\text{end}}}{2}$ . As they capture important spatio-temporal information and for simplicity, we only focus on these fundamental features in the following. For brevity, we write from now on just  $t$  and  $\Delta w$  instead of  $t_{i,j}$  and  $\Delta w_{i,j}$ , respectively, when the arc  $(i, j)$  is clear from the context. When analyzing the distributions of these features over all arcs of the training data as well as only on arcs appearing in solutions, it becomes apparent that they are highly skewed as illustrated by the histograms in Fig. 1. When performing logistic regression, this can be detrimental to model performance and result in biased predictions. We thus apply logarithmic transformations to obtain more balanced distributions and to reduce the impact of outliers, also see Fig. 1.

*ML-Model-Based Pruning.* As mentioned before, a best suited pruning rate for E-ADARP instances depends on the problem size and whole LNS configuration, in particular its allowed run time or number of iterations. We therefore consider the number of PD arcs to be preserved as an external input and will evaluate the pruning strategies for different values in our experiments. More specifically, to make this parameter less dependent on the actual instance size and directly comparable to parameter  $k$  from the  $k$ -NN approach, let us call this externally provided parameter also  $k$ , and the  $2nk$  PD arcs with highest heatmap values obtained from the model are preserved, the rest discarded. Note that the factor two comes from the fact that our  $k$ -NN approach keeps the best  $k$  in- as well as outgoing arcs.

*Issue of node degree imbalance.* As we will see in Section 5, the in-solution probability approximation works reasonably well using the two transformed features, but when corresponding models are applied for pruning within the LNS framework, the results disappoint compared to the baseline  $k$ -NN approach. A closer inspection revealed that the sparsening based on  $\log(t)$  and  $\log(|\Delta w|)$  led to a substantial imbalance in the remaining in- and outdegrees of the nodes. While some nodes still have a high number of incident edges, other nodes may lose connection even completely, leaving the LNS with too few options to integrate certain nodes into routes. We address this issue in two ways. On the one hand,



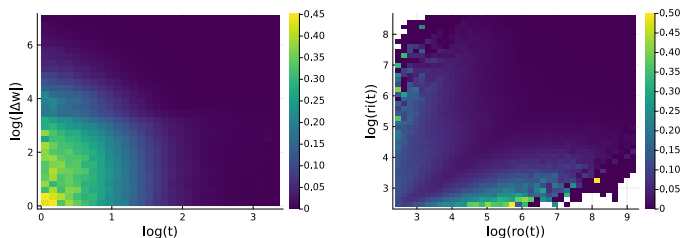


Fig. 2: Ratios of arcs with respective features to be part of a high quality solution.

we apply the  $k$ -NN sparsening but now use the heatmap values of the logistic regression as nearness criterion; in this way, a minimum in- and outdegree  $k$  is guaranteed while we still primarily rely on the predictions of the ML model. On the other hand, we introduce additional *rank-features*: The outgoing-rank  $ro(t)$  of an arc is the rank of the arc in the list of all outgoing arcs of the considered arc's source node sorted according to  $t$ . The rank-features  $ri(t)$ ,  $ro(\Delta w)$ , and  $ri(\Delta w)$  are correspondingly defined for ingoing arcs of the target node or in respect to  $\Delta w$  as sorting criterion, respectively. Observe that  $k$ -NN sparsening does nothing else but simply keeping arcs with ranks up to  $k$ . As these rank-features also have quite skewed distributions, we again apply logarithmic transformations, yielding the features  $\log(ro(t))$ ,  $\log(ri(t))$ ,  $\log(ro(\Delta w))$ , and  $\log(ri(\Delta w))$ .

*ML Models.* To determine which ML models to employ, we analyze the functions to be fitted with regards to the potential features. A visualization of ratios of arcs from the training data to appear in high quality solutions over two-dimensional feature spaces is shown in Fig. 2. While the illustrated functions are not linear, they appear not that hard to be approximated well. Considering this nonlinearity but also the requirement to handle large data sets and provide fast inference on large instances, we decided to investigate a support vector machine (SVM) [10] and gradient boosted trees (GBTs) [15]. As an SVM with a non-linear kernel could deal with the non-linear relationships of the features but would be too slow for the many samples we have to consider, we use a fast linear SVM enhanced with a Nyström transformer [29]. The latter approximates a radial-basis kernel via a low-rank matrix. The utilization of these two model types is further encouraged by their applicability for classification as well as regression, their broad usage, and often excellent results in many and also similar applications [see, e.g., 14, 16, 25]. We also deliberated using more advanced (graph) neural networks, however, considering concerns regarding scalability and model bias, we intentionally keep things relatively simple here and only present the above models.

## 5 Numerical Experiments and Results

All proposed approaches were implemented in Julia 1.11.5 including the GBT models via the Julia package EvoTrees, but for the linear SVM model with

Nyström transformer, we utilized Python’s scikit-learn library via the PythonCall interface. The experiments were executed on single cores of 2.4 GHz Intel Xeon E5-2640 v4 processors with a memory limit of 48 GB. The underlying solving approach for the E-ADARP is OTF-LNS [8], and we investigate the impact of the sparsification approaches on its scalability to the very large benchmark instances from Limmer [21], comparing the performance of the best resulting approaches to BI-LNS [21], the so far leading method on these instances. The five instances contain between 180 and 260 vehicles with the number of requests ranging from 3600 to 5200, and they follow the naming scheme  $an_H-n$ , where  $n_H$  denotes the number of vehicles and  $n$  the number of requests. According to the literature [7, 8, 21, 24], the objective function’s (1) weighting factors are set to  $w^{\text{routing}} = 0.75$  and  $w^{\text{excess}} = 0.25$ , the minimum end SoC  $\gamma$  is fixed at 0.7, and an arc’s battery consumption is proportional to the associated travel time. We used a time limit of 15 minutes per run, and 30 LNS runs were performed for each instance with each sparsification variant.

The training of the employed ML models is done on 100 independent instances of the same scale as the benchmark instances. For each size, 20 instances are randomly generated in the same way as detailed by Limmer [21]. Labels for training are derived from high-quality solutions of OTF-LNS by running it 30 times for six hours on each instance. Regarding the class imbalance arising from the substantial surplus of negative samples, we investigated random subsampling to obtain an equal number of positive and negative training samples. In preliminary studies, we compared models trained on this balanced data to models trained on the original data and found no substantial advantage. To avoid distortion of the data, we settled on using the original training data with the restriction to a randomly selected subset of 5% and 2.5% of the samples for the GBT and Ny-SVM models respectively to deal with the unmanageable size of the whole training data. Our 5% training set consisted then of around 94 million negative and 450,000 positive samples and roughly half as many remain in the 2.5% training set. After preliminary testing to find robust parametrizations, the configuration of the employed models is as follows: For the linear SVM, the penalty parameter  $C_{\text{svm}} = 100$ ,  $\gamma_{\text{svm}} = 1$ , and 30 landmarks are used in the Nyström transformer; all used features undergo standardization. For the GBT model, we set  $\eta = 0.1$ , the number of trees to 100, and a maximum depth of four, and we also apply monotonic decrease constraints on the travel time-based rank-features to indicate that an increase in value can be expected to always result in a decrease in the target value.

*Results.* As we are interested in the predictive performance of our employed models over a larger range of values for  $k$  and respective ratios of arcs to be pruned, we focus on the receiver operating characteristic (ROC) and precision-recall (PR) curves and their associated areas under the curve, AUC and AUPRC, as metrics. For brevity, we denote different subsets of features to train the models with as follows: Our main original features are  $F2 = \{\log(t), \log(|\Delta w|)\}$ , the rank-features derived from travel times are  $RF2 = \{\log(\text{ro}(t)), \log(\text{ri}(t))\}$ ,  $F4 = F2 \cup RF2$  is their combination, and  $RF4 = RF2 \cup \{\log(\text{ro}(\Delta w)), \log(\text{ri}(\Delta w))\}$  are all rank-

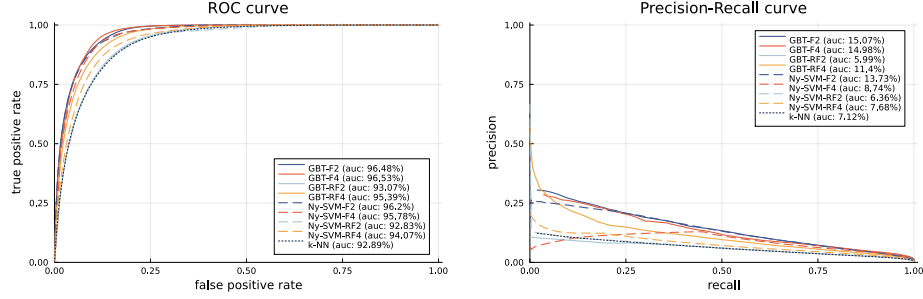


Fig. 3: ROC & PR curves of  $k$ -NN, GBT and Ny-SVM with various feature sets.

features. For evaluation of the proposed sparsening methods in combination with OTF-LNS, we compare the results to the best known solutions (BKSs) on the considered benchmark instances as reported by Limmer [21]. When it comes to testing the sparsification within the LNS, we consider  $k \in \{100, 150, 200, 250, 300, 350, 400, 500, 600, 800, 1000\}$  to see the impact of different pruning strengths.

The predictive performance of models GBT and Ny-SVM on the test data in comparison to the  $k$ -NN heuristic is illustrated in Fig. 3. The ROC curves and AUCs of all approaches across all feature sets indicate their capability of reasonably differentiating between positive and negative samples, whereas the worse results in terms of PR curves and AUPRCs reveal the impact of the unbalanced data. Overall, when the two ML models are performed with the same feature set, GBT consistently gives better results than Ny-SVM, but differences are very small. The used feature sets have generally more impact. In particular, we can clearly observe that the two original features in F2 lead to the best results. Interestingly, adding the time-based rank features (F4) slightly worsens results. Most importantly, the  $k$ -NN approach and models relying only on rank-features (RF2 and RF4) showcase considerably worse performance as indicated by their lowest AUC and AUPRC values.

We now evaluate the impact of the sparsification methods on the results of OTF-LNS in terms of the average percentage gap of the resulting objective values to the BKS over all instances; see Fig. 4. The original OTF-LNS without sparsification and BI-LNS are used as baselines as indicated by the red and black dashed lines in the plots, respectively. We can see that classical  $k$ -NN sparsification already significantly improves the solution quality. The only exception occurs for  $k = 100$ , where the pruning is too restrictive for OTF-LNS to find any feasible solutions. Perhaps surprisingly, GBT with F2 is not able to catch up with  $k$ -NN sparsening and the BKS. As already pointed out in Section 4, we explain this primarily by the aspect that too few incident arcs remain for some nodes, leaving the heuristic search too few options. In general, GBT and Ny-SVM achieve similar performances again when used with the same feature sets, as exemplified for feature set F2 in Fig. 4. Their best objective values are found for  $k = 150$  and  $k = 200$  respectively with %-gaps ranging from 0.5 to 1.4. Extending the

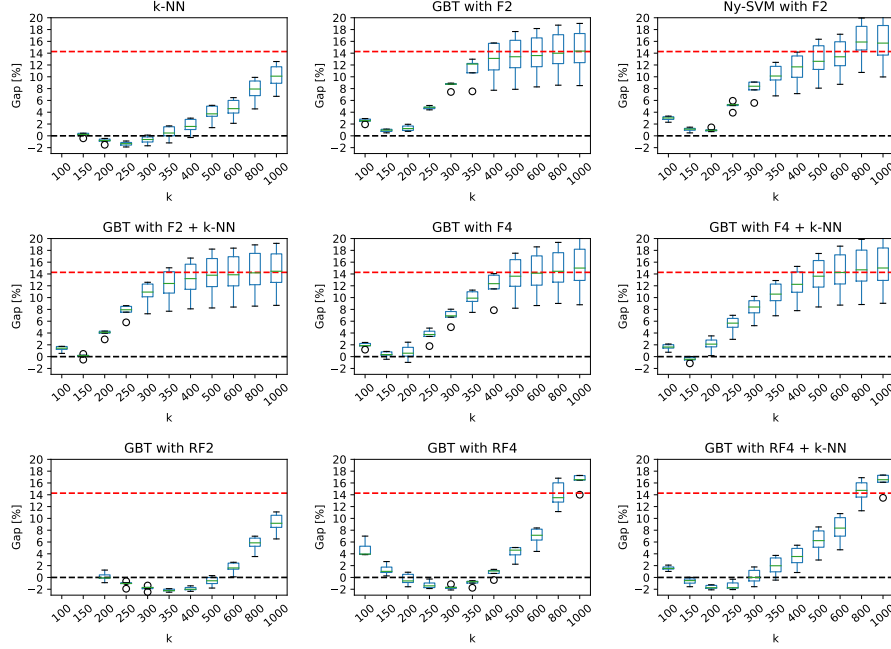


Fig. 4: Avg. %-gaps of objective values over all instances for OTF-LNS combined with various sparsening approaches considering different  $k$ -values. The dashed lines represent BI-LNS (black) and OTF-LNS (red).

feature set to F4 to also capture rank-information about arcs in relation to other arcs sharing a common node, improves the results only slightly, but also these variants are outperformed by the  $k$ -NN approach as well as BI-LNS.

Using the  $k$ -NN heuristic guided by the heatmap values to control the minimum node degree mitigates this issue and makes the approaches competitive with BI-LNS as shown in Fig. 4. This is especially beneficial for models using set F4 for which the peak performance is observed at  $k = 150$  with average %-gaps as low as  $-1.1\%$ .

As the basic  $k$ -NN heuristic still performs slightly better than our most promising models, we investigate the effect of putting more emphasis on the rank-features and employ models trained on sets RF2 and RF4. Even though these models perform worse in terms of predictions, their application during sparsification allows for larger improvements of the performance of OTF-LNS than with models of higher predictive power. Overall, RF2-based sparsification turns out to work best in the context of OTF-LNS with average %-gaps between  $-1.9\%$  and  $-2.5\%$  when using the GBT model with  $k = 350$ , thus outperforming all other investigated approaches as well as BI-LNS. The corresponding results on the benchmark instances in terms of the minimum and mean objective values ( $\text{obj}_{\min}$  and  $\overline{\text{obj}}$ ) as well as the standard deviation  $\sigma(\text{obj})$  are given in Table 1.

We observe that the best performing value for  $k$  is notably higher than for the other approaches and sparsification based on this feature set requires in general higher  $k$ -values in order to enable feasible solutions, as illustrated in Fig. 4. This needed decrease in the pruning rate can be explained by the reduced approximation capabilities of

RF2-based models as indicated by their according ROC and PR curves, and when utilizing other feature sets such as F4, significantly more in-solution arcs can be covered with less preserved arcs. We assume as major reason for this discrepancy that the models learn the properties of the average solution arcs but not those of the in general only comparatively few arcs that are likely really crucial for superior solutions, which might include ones with a low probability. For example, some arcs with long travel times may be needed to allow for sufficient charging opportunities. Thus, considering better approximation methods or more advanced features is unlikely to help as it is an inherent issue and limitation of the approach of covering as many arcs appearing in close-to-optimal solutions as possible, given a limited amount of arcs that may be kept in sparsening.

Table 1: Results on benchmark instances.

Instance	BI-LNS (Limmer, 2023)		OTF-LNS with GBT-RF2, $k = 350$		
	$\text{obj}_{\min}$	$\overline{\text{obj}}$	$\text{obj}_{\min}$	$\overline{\text{obj}}$	$\sigma(\text{obj})$
a180-3600	29177.96	29312.55	<b>28178.26</b>	28448.68	98.90
a200-4000	32013.56	32263.01	<b>31078.78</b>	31291.79	125.04
a220-4400	35259.06	35428.04	<b>34263.48</b>	34472.21	94.03
a240-4800	38270.98	38460.95	<b>37292.17</b>	37511.77	111.68
a260-5200	41472.11	41745.98	<b>40452.13</b>	40683.16	137.57

## 6 Conclusion and Future Work

We investigated methods for a heuristic graph sparsification in the context of the E-ADARP with the goal of increasing the scalability of OTF-LNS to very large-scale instances by reducing the search space substantially beforehand. As it is not obvious what distinguishes promising arcs from less promising ones, we explored the potential of ML models for this classification. The models are trained offline on representative instances and arcs appearing in close-to-optimal solutions are used as labels. We observed that both, a linear SVM with Nyström transformer as well as a GBT, are able to learn the average probabilities of arcs being part of high-quality solutions reasonably well when using travel times and differences in time window mid-points as features. The best approximation results as documented by ROC and PR curves are obtained for these features as well as their extension to rank-based features in respect to the travel times. Considering a common  $k$ -NN heuristic as baseline, we compare the performance of GBT and SVM models utilizing different subsets of features in the context of the OTF-LNS. We observe that while some feature combinations enable high predictive performance, this does not directly translate to OTF-LNS being able to find high quality solutions as, for example, crucial solution arcs might not fit the learned average properties of solution arcs due to the structural complexity of the problem. Thus, our investigation reveals a fundamental limitation of

such ML-based pruning approaches: superior predictive performance does not generally imply better solutions of the optimization. Enhancing the pure ML-based sparsification by employing the predictions as nearness criterion for the  $k$ -NN sparsening allows for controlling the node degrees and can lead to significant improvements depending on the employed features. Still, in the end we find that pure rank-based features in respect to travel times work best in our problem setting, allowing for substantial sparsification of large benchmark instances by a learned model and boosting the performance of OTF-LNS to beat the so far best solving approach, making it the new state-of-the-art on these instances.

Investigation of more advanced graph neural networks and their learning capabilities as by Sun and Yang [27] could be an interesting further line of research, but the literature is still scarce in this respect. Moreover, applying graph neural networks to dense graphs also does not come without issues when considering huge instances. As supervised learning based on exemplary solutions also has its limits, reinforcement learning is another promising direction to explore.

## References

1. Alanzi, E., Menai, M.E.B.: Solving the traveling salesman problem with machine learning: a review of recent advances and challenges. *Artificial Intelligence Review* **58**(9), 267 (2025)
2. Arnold, F., Gendreau, M., Sörensen, K.: Efficiently solving very large-scale routing problems. *Computers & Operations Research* **107**, 32–42 (2019)
3. Bellmore, M., Nemhauser, G.L.: The traveling salesman problem: A survey. *Operations Research* **16**(3), 538–558 (1968)
4. Bertsimas, D., Jaillet, P., Martin, S.: Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research* **67**(1), 143–162 (2019)
5. Bongiovanni, C., Geroliminis, N., Kaspi, M.: A ride time-oriented scheduling algorithm for dial-a-ride problems. *Computers & Operations Res.* **165**, 106588 (2024)
6. 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)
7. Bongiovanni, C., Kaspi, M., Geroliminis, N.: The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological* **122**, 436–456 (2019)
8. Bresich, M., Raidl, G., Limmer, S.: Letting a large neighborhood search for an electric dial-a-ride problem fly: On-the-fly charging station insertion. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 142–150, Association for Computing Machinery (2024)
9. Cordeau, J.F.: A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* **54**(3), 573–586 (2006)
10. Cortes, C., Vapnik, V.: Support-vector networks. *Machine Learning* **20**(3), 273–297 (1995)
11. Dumas, Y., Desrosiers, J., Soumis, F.: The pickup and delivery problem with time windows. *European Journal of Operational Research* **54**(1), 7–22 (1991)
12. Eppstein, D., Paterson, M.S., Yao, F.F.: On nearest-neighbor graphs. *Discrete & Computational Geometry* **17**(3), 263–282 (1997)
13. Fitzpatrick, J., Ajwani, D., Carroll, P.: Learning to sparsify travelling salesman problem instances. In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 410–426, Springer (2021)

14. Fitzpatrick, J., Ajwani, D., Carroll, P.: Learning to prune electric vehicle routing problems. In: Sellmann, M., Tierney, K. (eds.) *International Conference on Learning and Intelligent Optimization*, pp. 378–392, Springer (2023)
15. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* **29**(5), 1189–1232 (2001)
16. Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Curran Associates Inc. (2022)
17. Helsgaun, K.: An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research* **126**(1), 106–130 (2000)
18. Kool, W., van Hoof, H., Gromicho, J., Welling, M.: Deep policy dynamic programming for vehicle routing problems. In: Schaus, P. (ed.) *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 190–213, Springer (2022)
19. Lauri, J., Dutta, S.: Fine-grained search space classification for hard enumeration variants of subset problems. In: *AAAI Conference on Artificial Intelligence*, pp. 2314–2321, AAAI Press (2019)
20. Lauri, J., Dutta, S., Grassia, M., Ajwani, D.: Learning fine-grained search space pruning and heuristics for combinatorial optimization. *Journal of Heuristics* **29**(2), 313–347 (2023)
21. Limmer, S.: Bilevel large neighborhood search for the electric autonomous dial-a-ride problem. *Transportation Research Interdisciplinary Perspectives* **21**, 100876 (2023)
22. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. *Operations Research* **21**(2), 498–516 (1973)
23. Su, Y., Dupin, N., Parragh, S.N., Puchinger, J.: A branch-and-price algorithm for the electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological* **186**, 103011 (2024)
24. 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)
25. Sun, Y., Ernst, A., Li, X., Weiner, J.: Generalization of machine learning for problem reduction: a case study on travelling salesman problems. *OR Spectrum* **43**(3), 607–633 (2020)
26. Sun, Y., Li, X., Ernst, A.: Using statistical measures and machine learning for graph reduction to solve maximum weight clique problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43**(5), 1746–1760 (2021)
27. Sun, Z., Yang, Y.: Difusco: graph-based diffusion solvers for combinatorial optimization. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Curran Associates Inc. (2023)
28. Tian, H., Medya, S., Ye, W.: COMBHELPER: a neural approach to reduce search space for graph combinatorial problems. In: *AAAI Conference on Artificial Intelligence*, pp. 20812–20820, AAAI Press (2024)
29. Williams, C., Seeger, M.: Using the nyström method to speed up kernel machines. In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pp. 661–667, MIT Press (2000)
30. Zhang, S., Yang, Y., Tong, H., Yao, X.: Learning-based problem reduction for large-scale uncapacitated facility location problems. In: *IEEE Congress on Evolutionary Computation*, pp. 1–8, IEEE (2024)