hEART 2020 - 9th Symposium of the European Association for Research in Transportation

September 1-3, 2020, Lyon, France

A predictive large neighborhood search for the dynamic electric autonomous dial-a-ride problem

Claudia Bongiovanni [1] Mor Kaspi [2] Jean-François Cordeau [3]
Nikolas Geroliminis [1]

[1]Urban Transport Systems Laboratory, School of Architecture, Civil & Environmental Engineering
École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
{claudia.bongiovanni,nikolas.geroliminis}@epfl.ch
[2]Department of Industrial Engineering
Tel-Aviv University, Tel-Aviv 69978, Israel
{morkaspi}@tauex.tau.ac.il
[3]Department of Logistics and Operations Management
HEC Montréal, H3T2A7 Montréal, Canada
{jean-francois.cordeau}@hec.ca

ABSTRACT

The dynamic electric autonomous Dial-a-Ride Problem (e-ADARP) is a generalization of the dial-a-ride problem which employs electric Autonomous Vehicles (e-AVs) to provide shared rides to on-line requests. The goal of the dynamic e-ADARP is to maximize the number of served requests while minimizing operational cost and user excess ride time. To reach this goal, metaheuristics are designed to modify vehicle-trip assignments as information reveals over time. Differently from human-driven vehicles, e-AVs can be re-routed as often as desired in the course of operations. Given the on-line nature of the problem, plan modifications need to be efficiently performed to timely notify users and provide new instructions to the vehicles.

In this work, we present a new extension to the family of Large Neighborhood Search (LNS) metaheuristics, which employs a machine learning component to select destroy/repair couples from a pool of competing algorithms. At each iteration, the machine learning component predicts the objective function improvement that is expected to be obtained after the employment of each of the competing algorithms. The destroy/repair couple is consequently drawn according to the expected improvement proportions. Worsening solutions are also considered and drawn with the same likelihood of descent solutions. The proposed metaheuristic is denoted by Predictive Large Neighborhood Search (PLNS) and is employed to efficiently solve dynamic e-ADARP instances. Computational results are performed on 244 100-request dynamic instances from Uber Technologies Inc. Results show that PLNS outperforms the state-of-the art in the context of on-line operations.

Keywords: *dial-a-ride problem, on-line optimization, autonomous vehicles, metaheuristics, machine learning, random forests*

1. INTRODUCTION

In the Dial-a-Ride Problem (DARP), minimum cost routes and schedules are defined for a fleet of vehicles exiting known depots and serving a set of customers with given pickup and dropoff locations ([*14*]). The optimization can take into consideration multiple criteria and can include multiple type of users and destination depots (e.g. [*31*],[*9*]). Typical operational constraints include vehicle capacity, maximum route duration, and maximum user ride time constraints. In addition, service start times at pickup and dropoff locations are usually limited by time-window constraints. A recent work has extended the standard DARP by considering the use of electric autonomous vehicles (e-ADARP) ([*7*]).

The DARP literature can be divided into two main streams, namely static and dynamic DARP. In the first case, demand is fully known in advance, whereas, in the second case, demand is revealed on-line. No information about future requests is typically assumed, although some stochastic information (e.g. [*1*],[*23*]) and forecasts may be used (e.g. [*16*], [*17*])). Given the inherent uncertainty on demand, transportation requests are allowed to be denied in the dynamic DARP. Hence, solution quality is primarily measured by means of the total number of served requests and secondly by operational costs.

The use of electric autonomous vehicles (e-AVs) poses new challenges that need to be tackled on-line. That is, the planning process needs to continuously re-optimize the vehicle battery levels, decisions regarding detours to charging stations, recharging times, and destination depots together with the classic dial-a-ride features. The added complexity induced by the use of e-AVs for dynamic dial-a-ride systems calls for the design of new efficient heuristics that can be employed for real-time decision making.

Recent advance in artificial intelligence has fostered new research directions at the intersection between Machine Learning (ML) and optimization. While optimization methods have been often employed into ML algorithms (e.g. [*4*], [*22*], [*5*]), the use of ML into optimization algorithms is new and has been focusing on the following tasks: (1) ML as an approximation tool to tackle time-consuming tasks in discrete optimization algorithms ([*6*], [*25*], [*28*]), (2) ML to heuristically solve discrete optimization problems ([*26*], [*3*],[*36*]) (3) ML to choose among a number of competing algorithms to solve hard optimization problems ([*29*], [*37*], [*19*]). In this work, the machine learning component is employed in the context of a metaheuristic approach to predict the expected performance of each competing algorithms at each iteration of a LNS approach. Performance is measured by the expected objective function improvement. At each iteration, algorithms are drawn proportionally to their expected improvement. That is, the ML regression task replaces the "roulette wheel" mechanism proposed in the Adaptive Large Neighborhood Search (ALNS) ([*34*]) and is based on a statistics collected from a large dataset of examples of moves from all of the competing LNS algorithms on several instances and routing solutions.

In summary, this work proposes a Predictive Large Neighborhood Search (PLNS) heuristic to solve the dynamic electric Autonomous Dial-a-Ride Problem (e-ADARP). The scope of the paper is to provide the following contributions: (1) an efficient metaheuristic to deal with on-line vehicle routing problems, (2) show that the use of machine-learning within optimization algorithms leads to positive results and fosters new research directions, (3) compare the proposed approach against the state of the art (i.e. ALNS), (4) train and test the PLNS with a large dataset obtained by simulations from real transportation data from Uber Technologies Inc.

2. SOLUTION APPROACH

Heuristic two-phase approaches have been widely applied to a variety of tightly-constrained vehicle routing problems (e.g. [*2*], [*35*], [*32*]). The first phase typically attempts to assign requests to vehicles through a polynomial-time insertion heuristic ([*24*]). The assignment needs to be feasible and thus respect all of the constraints imposed by the problem. In the e-ADARP, such constraints correspond to time-window, capacity, maximum ride time, and battery constraints.

The second phase is designed to re-optimize previous decisions through sequential inter- and intra-vehicle request exchanges and charging plan modifications. The route adjustments are performed by sequentially destroying and repairing part of the vehicle plans, thus following a LNS approach ([*33*]). The destruction degree is typically drawn from a uniform distribution supported on a bounded interval. If the bound is too restrictive, the incumbent solution may be modified only marginally. As a result, the search may have difficulties in moving towards promising neighborhoods and get trapped into a local minimum. One way of dealing with this drawback is to loosen the interval bound, allowing the algorithm to rearrange a higher percentage of the vehicle requests. Nevertheless, higher destruction degrees typically have a negative effect on computational time ([*21*],[*27*],[*34*]). Another methodology which is typically employed to try and escape local minima explores non-improving solutions (e.g. [*13*],[*15*]). Worsening solutions may be generally explored by employing an acceptance criterion based on Simulated Annealing (SA) ([*30*]). If non-improving solutions are accepted, the solution is expected to iteratively recover towards promising areas of the search space. Such recovery cannot be guaranteed as it highly depends on the type of operator and destruction degree being employed. Thus there is a maximal number of successive non-improving steps allowed, which may determine an early exit from the search. The state of the art (i.e. ALNS) selects operators according to a score function providing a measure for their success on previous iterations (e.g. [*18*], [*34*]). The score function is initialized according to a "roulette wheel" mechanism and may need several iterations before being able to choose between the competing operators. Indeed, ALNS is typically employed for large static problems, which are not as tightly bound by computational time as for dynamic settings.

In this work, we propose another LNS extension, i.e. the PLNS,in which the expected percentage improvement for each competing algorithm is predicted at any given iteration. The operator to be used is then drawn according to the relative improvement ratios. The advantage of the method is that the scores do not need to be adjusted through several iterations, as in the ALNS. Instead, they are spontaneously provided for each sub-problem during the LNS search. The following sections introduce the machine-learning approximation, present the PLNS, and the construction of the training dataset in more detail.

3. MACHINE-LEARNING APPROXIMATION

Consider a problem instance i and an incumbent solution s characterized by aggregated features (i.e. input variables) $\{X_1, \ldots, X_d\}$. Such features may correspond to scalar, boolean, or more generally categorical values (e.g. number of requests in the vehicle future plans, the presence of a new upcoming request, current vehicle tasks etc.). The d-dimensional vector $\mathbf{x}_i = \{x_{i1}, \ldots, x_{id}\}$ denotes one realization of such features and defines a datapoint. Tuples $(\mathbf{x}_i, y_i)$, with $i = \{1, \ldots, N\}$, are a collection of examples of problems for which the output y_i (e.g. the most frequent percentage improvement) is known, i.e. the labeled set. Sample an arbitrary large subset of examples within the labeled set, called the training set with size N_{train}. A supervised learning approach aims at estimating a function $\xi : \mathcal{X} \rightarrow \mathcal{Y}$ which best maps the input space to the output space through the examples provided by the training dataset. If $\mathcal{Y} \in \mathbb{R}$, ξ is a regressor (e.g. it estimates the most frequent percentage improvement). The goal of the regression problem is to minimize a loss function $\mathcal{L}$ measuring the discrepancy between the predicted and known output values from the test dataset, with size N_{test}. The test dataset is the examples that remain from the labeled set that are not in the training set. There exists multiple measures providing the expected prediction error. For a regression problem, model accuracy can be estimated by employing measures such as: (1) Mean absolute error (MAE, also known as L-1 loss), (2) Root mean square error (RMSE, the root of the MSE or L-2 loss), (3) Coefficient of determination (R^2). The three measures are summarized here below:

$$\text{MAE} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} |y_i - y_i{}^p|$$

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (y_i - y_i{}^p)^2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N_{\text{test}}} (y_i - y_i{}^p)^2}{\sum_{i=1}^{N_{\text{test}}} (y_i - \bar{y_i})^2}$$

Where y_i denotes the observed data, $y_i{}^p$ the predicted data, and $\bar{y_i}$ the mean of the observed data. For a review exploring the meaning and differences between these measures for machine larning applications, the reader is refferd to the on-line website in [*20*].

In this work, the prediction problem is tackled through ensemble learning, namely Random Forest (RF) regression ([*11*]). The goal of ensemble learning is to combine simple and fast learners to obtain better performance. In the case of RF, the weak learners are Classification And Regression Trees (CART) ([*12*]). CART recursively partition the input space through a series of binary splits which are chosen through a mathematical model which maximizes a goodness of split function. The depth of the tree is typically controlled by a parameter which provides an upper bound on the maximal number of splits and, consequently, sub-regions. For a regression problem, the value of each sub-region is decided by computing the average of the samples in the sub-region (the "average vote"). The predicted value for a new input point is then obtained by passing the point through the tree until a final node (or sub-region) is reached. In order to increase the accuracy of the prediction, multiple decision trees are typically combined through a technique called *bagging*, which essentially generates multiple models based on bootstrap samples of the input space ([*10*]). Random forests attempt to further increase the accuracy of the predicted models and de-correlate trees by choosing a subset of the feature space at every split in the tree. The final prediction is the aggregation of the predictions of all models. For a regression task, the aggregation corresponds to the average of the predicted values. Random forests have a similar performance compared to other popular machine-learning methodologies, such as support vector machines, while having numerous practical advantages. What makes RF most suitable for this work is its ability to automatically select features, provide a variable importance measure, handle outliers and very big datasets.

4. PREDICTIVE LARGE NEIGHBORHOOD SEARCH

This section describes the predictive LNS approach proposed for the e-ADARP, i.e. the PLNS. The PLNS differs in multiple ways with respect to the ALNS extension proposed in [*34*]: (1) The PLNS concurrently selects destroy/repair operators to be employed at any new iteration during the search, (2) The selection of the operators are guided by a machine-learning approach, (3) The prediction scheme selects the operators in consideration of the expected objective improvement on the problem instance and routing solution at the current LNS iteration, (4) The repair operators use the exact scheduling and battery management algorithm proposed in [*8*].

The destroy and repair operators proposed in this work closely follow the ones proposed in [*34*]. Namely, this work adopts all of their destroy (i.e. shaw, worst, and random) and repair (i.e. basic greedy, regret-2, and regret-3) heuristics. Consider an instance $i \in \{\text{instances}\}$, characterized by planned transportation requests and a new transportation requests j. The current routing solution $s \in \{\text{solutions}\}$ has total operational cost $f(s)$, which includes a high penalty for the unassigned request j. If more than one new transportation request appear at the same time, the requests are treated independently and in the order of arrival. The attempt of the PLNS is to insert j, while reducing the total operational cost $f(s)$. In order to achieve this goal, the PLNS destroys and repairs the current routing solution over multiple iterations $i \in \{1, \ldots, k\}$. At each iteration, the search uses a specific destroy operator $d(i)$ and repair operator $r(i)$. Destroy-Repair couples $(d(i), r(i))$ are selected according to the process explained next. First, the destroy level q_i is drawn from a uniform bounded interval. The destroy level, instance, and routing solution information forms the input data $\mathbf{x}_i = \{x_{i1}, \ldots, x_{id}\}$ which is fed to the destroy-repair regression models $\xi_{(d_l, r_n)}$, with $l, n \in \{1, \ldots, m\}$. The regression models return the expected performance of each destroy/repair couple at the current iteration i, i.e. the most frequent objective improvement $y(d_l, r_n)_i$. Successively, the predicted performances are normalized returning the proportion of improvement per destroy/repair couple $p(y(d_l, r_n)_i)$ at iteration i. Note that negative improvements are also possible and are drawn with equal probability of positive improvements, given their relatively low magnitude. Successively, the destroy-repair couple to be employed at the current LNS iteration, i.e. $(d(i), r(i))$, is drawn according to the predicted proportions $p(y(d_l, r_n)_i)$. The destroy-repair couple $(d(i), r(i))$ selects q_i requests to be removed from s and update the set of pending requests to $q_i \cup \{j\}$. Define $s^{'}$ as the new solution obtained by destroying the pending requests from s through $d(i)$ and repairing them through $r(i)$. If $s^{'}$ is feasible, contains all of the removed requests, including the new request j, and reduces total cost $f(s)$, the incumbent solution s is updated. The acceptance criterion, based on SA, allows the search to explore non-improving solutions $s^{'}$. Note that the search is prematurely terminated if the maximal number of sequential non-improving iterations is exceeded and if a maximum time limit is exceeded (e.g. 5 seconds). Algorithm 1 provides a pseudo-code for the proposed PLNS framework.

6. DATASET, LABELING AND FEATURE SELECTION

This section presents the ground truth data generation (the creation of examples, also called the labeling), and the feature selection used for the PLNS. The prediction scheme for the dynamic e-ADARP uses of a large dataset containing tuples $(\mathbf{x}_i, y_i)$ to determine the percentage improvement to be expected by the use of destroy/repair operators (d_l, r_n), with $l, n \in \{1, \ldots, m\}$, at iteration i. While in most machine learning applications the output values are known, in the dynamic e-ADARP the labels have to be produced through extensive optimization-based simulations.

Algorithm 1: PLNS heuristic

```
Input: instance $i \in \{instances\}$, current solution $s \in \{solutions\}$, $j \in \{newrequests\}$, number of LNS iterations $k$, $m$ destroy
    models $d_l$, $m$ repair models $r_n$, regression models $\xi_{(d_l,r_n)}$ for $l, n \in \{1, \ldots, m\}$, LNS exit rule exit
Output: $s_{\text{best}}$
1  Initialize: $s_{\text{best}} = s$;
2  Initialize: $s_{\text{current}} = s$;
3  Initialize: check = true;
4  Initialize: penalty = Inf;
5  for $i \leftarrow 1{:}k$ do
6      Draw: $q_i \in \mathbb{N}$;
7      Compute: $\mathbf{x}_i$ from $i$, $s_{\text{current}}$, and $q_i$;
8      Predict: $\xi_{(d_l,r_n)} : \mathbf{x}_i \to y(d_l, r_n)_i$ for $l, n \in \{1, \ldots, m\}$;
9      Compute: $p(y(d_l, r_n)_i) = \frac{y(d_l,r_n)_i}{\sum_{l=1}^{m} \sum_{n=1}^{m} y(d_l,r_n)_i}$ [%];
10     Draw: $p(y(d_l, r_n)_i) \to (d(i), r(i))$;
11     Compute: $s^{'} = d(i)(r(i)(q_i \cup \{j\}))$;
12     if $s^{'}$ is infeasible then
13         check = false;
14     else
15         check = true;
16     if check = true then
17         if $f(s^{'}) < f(s_{best})$ + penalty then
18             $s_{\text{best}} = s^{'}$
19             $s_{\text{current}} = s^{'}$
20             $j = \emptyset$
21             penalty = 0;
22         else if accept($s^{'}$, $s_{current}$) is true then
23             $s_{\text{current}} = s^{'}$;
24     if exit = true then
25         break;
```

Several instances are simulated through an event-based simulation framework. The simulation includes vehicle events (such as arrivals, departures, recharges) and requests events (such as arrivals, pickups, and dropoffs). Insertions are triggered any time a request arrival event appears. The re-optimization through PLNS is triggered any time an upcoming request j cannot be directly inserted into the vehicle plans.

Consider an initial sub-problem i, characterized by routing solution s_{current} and cost $f(s_{\text{current}})$. The initial sub-problem may be adjusted by applying any of the available destroy-repair operators (d_l, r_n) with $d_l \in \{1 : \text{Worst}, 2 : \text{Random}, 3 : \text{Shaw}\}$ and $r_n \in \{1 : \text{Greedy}, 2 : \text{Regret2}, 3 : \text{Regret3}\}$. Since the destruction level q_i is randomly drawn from a uniform bounded interval, each destroy-repair operator (d_l, r_n) is applied to s_{current} for N times. Ideally, N should be large enough in the attempt to de-randomize the performance of each operator. For each trial and every operator (d_l, r_n), we retrieve the percentage improvement on the objective of the current solution from the application of operators (d_l, r_n). Note that the percentage improvement may be negative and that solutions that increase the objective function too much are discarded by a SA approach. After the N trials, we determine distributions on the performance of each destroy/repair couple (d_l, r_n). In particular, we retrieve the mode of the distributions (i.e. the most frequent objective improvent) and the number of samples in the distribution. If a distribution contains less than $p\%$ samples (i.e. $(1 - p\%)$ trials with operators (d_l, r_n) were discarded by the SA approach), the distribution is discarded from the training dataset. Note that if no destroy/repair operator is able to produce dis-

TABLE 1 Features description

Requests in vehicle plans	**New request**
Problem size	Is there a new request to be inserted? (binary)
Radius (average/std distance between requests and the centroid)	Is the new demand in the convex hull of the requests in the plan? (binary)
Area covered by the requests	Average distance of the new request from the centroid
Percentage of drop-offs in the convex hull from the pickups	Average distance of the drop-off of the new request from all nodes
Percentage of pickups in the convex hull of the drop-offs	
Vehicles	**Vehicle Routes**
Average/std distance of vehicles from the centroid	Time elapsed from the beginning of the planning horizon
Average distance between vehicles	Percentage of non-utilised vehicles
Average/std distance between vehicles and the pickup of the new request	Percentage of currently empty vehicles
Percentage of vehicles in the convex hull of pickups/dropoffs	Average/std current and future vehicle loads
LNS iterations	Average/std number of requests in the vehicle routes
SA temperature	Average/std vehicle travel times and requests excess times
Percentage cost difference from the initial/last incumbent solution	Average/std vehicle batteries
Neighborhood size (destroy level)	Average/std travel times from vehicle plans and unique travel times (cost matrix)
Percentage of non-improving iterations	Average/std vehicle tour lengths and number of edges

tribution of at least $p\%$ samples, then the search is exited. The modes of each destroy/repair couple (d_l, r_n) are successively retrieved and normalized. The best destroy/repair couple $(d(i), r(i))$ for labeling sub-problem i is finally drawn from the resulting mode proportions.
Throughout the simulations, we characterize instances i by several aggregated features which relate to: (1) The spatial distribution of requests in the vehicle plans, (2) The spatial distribution of vehicles and their routes; (3) Information on the upcoming request, (4) Information on current and past LNS iterations. Overall, we determine 40 features, summarized in Table 1.

7. NUMERICAL EXPERIMENTS

This work employs real data from Uber ride-shares in San Francisco, obtained by extracting pickup/dropoff locations and times from published GPS logs.[1]. The dataset results in 24,400 Uber trips in a one-week time interval. For the sake of our tests, we extract 100-request scenarios. Note that larger problems are not necessarily more difficult to solve, as complexity depends on the size of the encountered sub-problems, which in turn depend on the demand distribution and arrival rate, rather than the size of the scenarios. We construct a 15-minutes time-window around the request pickup/dropoff times and assume booking times happen at the beginning of the time-windows. We consider a fixed fleet size of 10 vehicles, with a maximal capacity of 15 passengers and a nominal battery capacity of 14.85 kWh. The fleet size has been chosen such that 65% of the total demand can be served by a greedy insertion algorithm, on average.

The procedure described in section 6 was applied to each one of the 244 100-request instances from the Uber dataset. At each LNS iteration, we employed each of the 9 LNS algorithms 200 times and discretized the destroy degree into four bounded distributions, up to 55%. That is, ech LNS algorithm tries to destroy up to 15%, 25%,35%, and 55% of the vehicle routes, 50 times each. Finally, we gathered information for all distributions composed of at least 50 samples (or 25% of 200 trials). The simulations resulted in a total of 20,449 labeled datapoints. All of the results were derived from a commercial 2.5 GHz quad-core computer with 16 GB of RAM and are implemented in the julia computing language.

The first question that is to be asked in the context of an online algorithm is whether the re-optimization policy actually increases the number of accepted requests. In fact, given that we do not estimate future requests arrivals, optimizing vehicle routes at any given time may decrease chances to accept requests later on. In Figure 1(a), we provide the difference in terms of the

[1]Available at https://github.com/dima42/uber-gps-analysis/tree/master/gpsdata

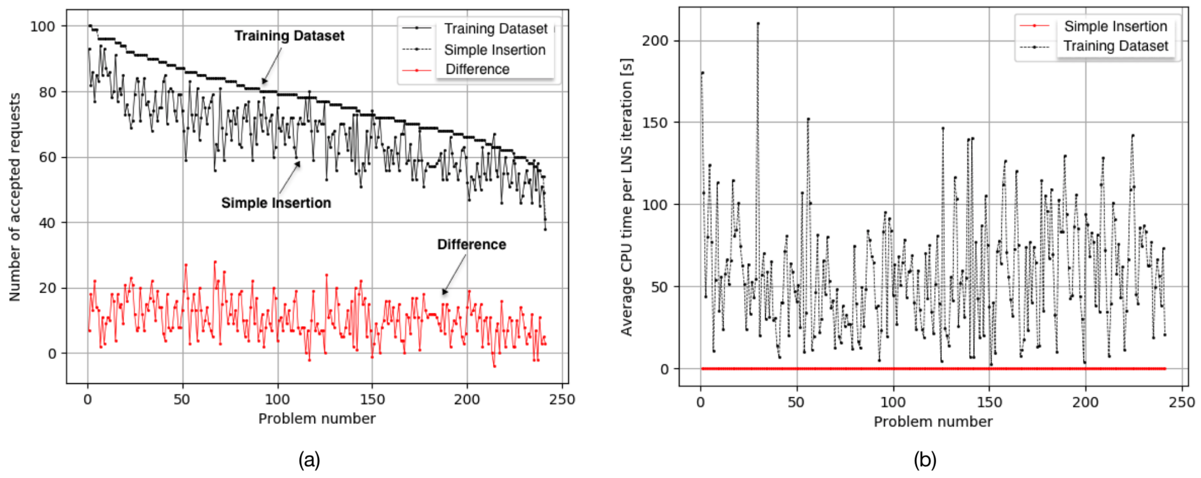

FIGURE 1 Training dataset and simple insertion: (a) Difference in number of accepted requests, (b) Average CPU time at each iteration of the training process

number of accepted requests between a simple insertion strategy and the training dataset. As it can be noted, re-optimizing routes may sensibly increase the number of accepted requests and by up to 25%. In a very few cases, the re-optimization phase may not lead to a higher number of accepted requests and may instead slightly decrease the solution quality (by less than 5%). Clearly, given its computational burden, the extensive methodology used to produce the training dataset is not suitable for real-time decision processes, as shown in Figure 1(b). For this reason, the re-optimization policy is learnt through a machine-learning approach and efficiently re-applied to the problem instances.

7.1 ML results

A random forest regressor is trained on 20,242 datapoints by the use of the python ScikitLearn library. The number of labeled datapoints per class (i.e. the most likely percentage improvement per LNS algorithm) is provided in Table 2. Note that the dataset is unbalanced, given that in training set each next move is drawn in consideration of the estimated improvement distributions. As customary in the ML literature, 75% of the labeled set is selected for training, and the remaining 25% for testing. The split is performed randomly. On the training dataset, we optimize the hyperparameters using grid search and k-fold cross-validation. The optimized hyperparameters include the number of estimators, maximal depth, maximal number of features, and minimal number of samples per split in the RF. The optimal parameters are chosen from the results obtained on 5 folds and by considering the L-2 loss. Table 3 shows performance measures obtained for each of the 9 trained regression models by applying them on the test dataset. As it can be noticed from the MAE and RMSE, the models predict the improvements that should be obtained by the application of any of the 9 algorithms within a 2% error, on average. Relative to the mean of the observed data (rRMSE), this translates to an error of less than 19%. All of the proposed models are characterized by a coefficient of determination (R^2) higher than 99%, showing that the regression models can replicate the outcomes with high confidence.

TABLE 2 Number of datapoints per class

Class	Number of datapoints
1	1,266
2	2,497
3	2,652
4	2,033
5	2,971
6	3,217
7	1,110
8	2,192
9	2,304

TABLE 3 Performance measures obtained for the RF regressor on the test dataset

Class	MAE	RMSE	R^2	rRMSE
1	1.42	2.13	99.71%	9.20%
2	1.15	1.75	99.65%	7.80%
3	1.12	1.66	99.70%	10.83%
4	1.95	2.64	99.28%	16.83%
5	1.38	2.04	99.46%	16.37%
6	1.39	2.07	99.41%	18.25%
7	2.15	3.27	99.10%	15.90%%
8	1.80	2.65	99.17%	18.94%
9	1.84	2.63	99.18%	18.88%

7.2 PLNS results

The trained models are re-applied on 30 out of the 60 100-request instances from the test dataset and in the context of the PLNS described in section 4. The PLNS approach is compared to a random selection algorithm and to ALNS, according to the score updating methodology proposed Ropke and Pisinger (2006). All of the algorithms are tested 10 times on each scenario in the test dataset. The maximal number of LNS iterations is set to 100 and the search is either exited when the search cannot recover after 10 consecutive worsening iterations or when a maximal time limit of 5 seconds is exceeded. The SA parameters, notably the initial temperature and the cooling rate, as well as the score adjustments parameters for the ALNS, are set as in Ropke and Pisinger (2006). However, we set the ALNS reaction factor r to 0.4 and update the operators scores every 5 iterations. That is, operators weights are updated more frequently and react faster to changes in the effectiveness of the algorithms.

Figure 2 compares ALNS, the random algorithm, and PLNS in terms of the number of accepted requests (i.e. our primary objective). As it can be noted, in most of the instances PLNS outperforms the two other approaches, on average. In some cases, PLNS may clearly outperform the other two algorithms and by more than 15-20%, e.g. problems 12, 20, 21, 24, 25. There are only a few cases in which the three algorithms perform similarly, e.g. problems 13, 15, and 28. There is only one case, i.e. problem 30, in which ALNS outperforms PLNS by only 5%. Finally, PLNS is able to increase the number of accepted requests by up to about 25% with respect to a simple insertion strategy.

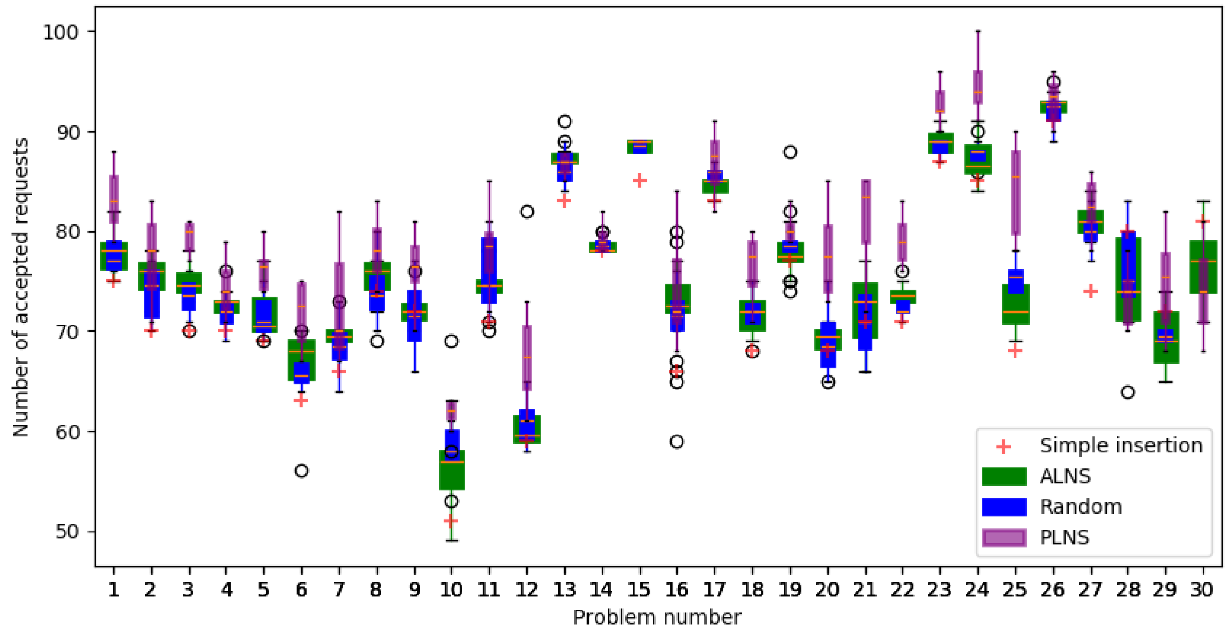

FIGURE 2 Distributions on the number of accepted requests on the test dataset

CONCLUSION

This work proposes a new extension to the family of large neighborhood search metaheuristics for the dynamic electric autonomous dial-a-ride problem (PLNS). The proposed metaheuristic employs a machine learning approach to predict the performance on the objective function of 9 destroy-repair couples for each iteration during the search. The training dataset is composed of a large number of examples of LNS moves and their respective expected objective improvement. We have trained 9 different models through a random forest regression approach. Results show that the learnt models are capable of reproducing the observed data with high confidence. The models have been re-employed on the test dataset in the context of a simulation-based optimization approach. Results show that PLNS outperforms the state of the art (ALNS) in terms of the number of accepted requests by making more informative neighoborhood moves. Current work is focusing on extending computational results and extracting managerial insights from the new optimization policy. Future work may include: (1) The adoption of a look-ahead policy to partially guide the optimization algorithm by future demand arrivals, (2) The adoption of other classification frameworks from machine learning literature.

ACKNOWLEDGMENTS

The authors would like to thank Dan Assouline, Tim Hillel, Gael Lederrey, and Nicholas Molyneaux for the several brainstorming talks about machine learning and its applications in the context of our proposed work. The authors would also like to thank George Adaimi for helping on the setup for the HPC facilities at EPFL.

REFERENCES

1. Albareda-Sambola, M., Fernández, E., Laporte, G., 2014. The dynamic multiperiod vehicle routing problem with probabilistic information. Computers & Operations Research 48, 31–39.
2. Archetti, C., Fernández, E., Huerta-Muñoz, D.L., 2018. A two-phase solution algorithm for the flexible periodic vehicle routing problem. Computers & Operations Research 99, 27–37.
3. Bengio, Y., Lodi, A., Prouvost, A., 2018. Machine learning for combinatorial optimization: a methodological tour d'horizon. arXiv preprint arXiv:1811.06128 .
4. Bertsimas, D., Dunn, J., 2017. Optimal classification trees. Machine Learning 106, 1039–1082.
5. Bertsimas, D., Shioda, R., 2007. Classification and regression via integer optimization. Operations Research 55, 252–271.
6. Bonami, P., Lodi, A., Zarpellon, G., 2018. Learning a classification of mixed-integer quadratic programming problems, in: International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Springer. pp. 595–604.
7. Bongiovanni, C., Kaspi, M., Geroliminis, N., 2019. The electric autonomous dial-a-ride problem. Transportation Research Part B: Methodological 122, 436–456.
8. Bongiovanni, C., Kaspi, M., Geroliminis, N., 2020. An exact scheduling and battery management algorithm for the electric autonomous dial-a-ride problem. Working paper.
9. Braekers, K., Caris, A., Janssens, G.K., 2014. Exact and meta-heuristics approach for a general heterogeneous dial-a-ride problem with multi depots. Transportation Research Part B: Methodological 67, 166–186.
10. Breiman, L., 1996. Bagging predictors. Machine learning 24, 123–140.
11. Breiman, L., 2001. Random forests. Machine learning 45, 5–32.
12. Breiman, L., 2017. Classification and regression trees. Routledge.
13. Cordeau, J.F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37(6), 579–594.
14. Cordeau, J.F., Laporte, G., 2007. The dial-a-ride probelm: models and algorithms. Annals of Operations Research 153, 29–46.
15. Cordeau, J.F., Laporte, G., Mercier, A., 2001. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational research society 52, 928–936.
16. Ferrucci, F., Bock, S., 2016. Pro-active real-time routing in applications with multiple request patterns. European Journal of Operational Research 253, 356–371.
17. Ferrucci, F., Bock, S., Gendreau, M., 2013. A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods. European Journal of Operational Research 225, 130–141.
18. Goeke, D., Schneider, M., 2015. Routing a mixed fleet of electric and conventional vehicles. European Journal of Operational Research 245, 81–99.
19. Gomes, C.P., Selman, B., 2001. Algorithm portfolios. Artificial Intelligence 126, 43–62.
20. Grover, P., 2018. 5 regression loss functions all machine learners should know: Choosing the right loss function for fitting a model. URL: `https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learne`
21. Gschwind, T., Drexl, M., 2019. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. Transportation Science 53, 480–491.

22. Gunluk, O., Kalagnanam, J., Menickelly, M., Scheinberg, K., 2016. Optimal generalized decision trees via integer programming. arXiv preprint arXiv:1612.03225 .
23. Ichoua, S., Gendreau, M., Potvin, J.Y., 2006. Exploiting knowledge about future demands for real-time vehicle dispatching. Transportation Science 40(2), 211–225.
24. Jaw, J.J., Odoni, A.R., Psaraftis, H.N., Wilson, N.H., 1986. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research Part B: Methodological 20, 243–257.
25. Kruber, M., Lübbecke, M.E., Parmentier, A., 2017. Learning when to use a decomposition, in: International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer. pp. 202–210.
26. Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., Lodi, A., 2019. Predicting tactical solutions to operational planning problems under imperfect information. arXiv preprint arXiv:1901.07935 .
27. Li, B., Krushinsky, D., Van Woensel, T., Reijers, H.A., 2016. An adaptive large neighborhood search heuristic for the share-a-ride problem. Computers & Operations Research 66, 170–180.
28. Lodi, A., Zarpellon, G., 2017. On learning and branching: a survey. Top 25, 207–236.
29. Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M., 2013. Algorithm portfolios based on cost-sensitive hierarchical clustering, in: Twenty-Third International Joint Conference on Artificial Intelligence.
30. Nikolaev, A.G., Jacobson, S.H., 2010. Simulated annealing, in: Handbook of metaheuristics. Springer, pp. 1–39.
31. Parragh, S.N., 2011. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Transportation Research Part C: Emerging Technologies 19(5), 912–930.
32. Parragh, S.N., Doerner, K.F., Hartl, R.F., Gandibleux, X., 2009. A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. Networks: An International Journal 54, 227–242.
33. Pisinger, D., Ropke, S., 2010. Large neighborhood search, in: Handbook of metaheuristics. Springer, pp. 399–419.
34. Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation science 40, 455–472.
35. Schilde, M., Doerner, K.F., Hartl, R.F., 2011. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. Computers & operations research 38, 1719–1730.
36. Vinyals, O., Fortunato, M., Jaitly, N., 2015. Pointer networks, in: Advances in Neural Information Processing Systems, pp. 2692–2700.
37. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K., 2008. Satzilla: portfolio-based algorithm selection for sat. Journal of artificial intelligence research 32, 565–606.