Simulation-based Evaluation of Dynamic Vehicle Routing Problem Features for Algorithm Selection

Thomas Mayer*, Tobias Uhlig, Oliver Rose

Institute of Computer Engineering, Universität der Bundeswehr München, Werner-Heisenberg-Weg 39, 85579 Neubiberg, Germany; *thomas.mayer@unibw.de

SNE 29(4), 2019, 179-188, DOI: 10.11128/sne.29.tn.10493
Received: May 10, 2019 (Selected ASIM SST Hamburg 2018 Postconf. Publ.), Accepted: Sept. 10, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Algorithm selection based on problem instance features is a common method to choose approaches for NP-hard combinatorial optimization problems. Features concerning the location of nodes for the Vehicle Routing Problem (VRP) are intensively studied. Although the dynamic version of the problem is gaining steadily in importance, there is a lack in research targeting problem features describing the dynamism of an instance. In our paper we translate known VRP features to features for the Dynamic Vehicle Routing Problem (DVRP). We investigate the suitability of them for algorithm selection. To this end, we explore the performance space of a Greedy and a Re-planning algorithm for various dynamic problem instances using simulation and an evolutionary algorithm. For the algorithm selection we investigate our features in combination with a supervised machine learning technique. The applicability of our features is demonstrated in a use case for autonomous algorithm selection of DVRP instances.

Introduction

In the last years, the research interest in vehicle routing has significantly increased due to our modern way of life. We order different goods of daily use over the Internet, we share cars and bicycles and use on-demand transportation services. To provide all these services, complex Vehicle Routing Problems (VRPs) with several constraints have to be managed and solved. The fundamental VRP was introduced by Dantzig and Ramser in [1] as generalization of the Traveling Salesman Problem (TSP) introduced from Flood in [2]. Essentially, it encompasses the planning of routes for vehicles to satisfy customer demands. The routes usually start and end at a central depot. A current survey and taxonomy on the VRP is, for example, [3]. Especially, the advances of information and communication technologies and the changing nature of the real-world problem instances put the dynamic version of the VRP into the research focus. Pillac et al. in [4] and Psaraftis et al. in [5] describe an explosion in the number of related papers after the year 2000. The Dynamic Vehicle Routing Problem (DVRP) was first introduced from Wilson and Colvin in [6], with a description of a computer controlled Dial-A-Ride system in Rochester, NY (USA). The DVRP is an extension of the VRP and is characterised by changing problem parameters over the time, [4], for example, additional customer demands occur during route execution. A current survey on the DVRP is provided in [7]. As a result of the increased research interest, an increasing number of different solution approaches to tackle the DVRP are available. Psaraftis et al. identifies the five main solution methods: *Tabu Search*, *Various Neighborhood Search*, *Insertion Methods*, *Dynamic Programming*, and *Markov Decision Process* in [5]. For each method several different heuristics and implementations exist. Simulation is also commonly used to handle the dynamic and stochastic aspects of vehicle routing. For example Juan et al. improves the Clarke and Wright heuristic, [8], by Monte Carlo Simulation, [9]. Juan et al. provide a review of methods using simulation to solve combinatorial optimization problems in [10]. The concept of Simheuristic is introduced, a methodology that integrates simulation into the solution finding process for combinatorial optimisation problems. [11] use this methodology to solve the Two-Echelon Location Routing Problem, a combination of the Capacitated Location Routing Problem (CLRP) and a VRP. Due to the intense research in this area and the increased number of algorithms and solution approaches for different variations of the

DVRP, the following research questions arise: Which is the best solution approach or algorithm for a specific problem instance? And, is it possible to distinguish between problem instances to select the algorithm that has a better expected performance? With regard to the No Free Lunch Theorem introduced in [12], we know that there is no strictly superior heuristic that outperforms all other heuristics for all problem instances. That means some problem instances are more difficult or harder to solve for certain algorithms compare to others. Our research focuses on learning which problem instances are easier or harder for a certain algorithm based on problem features. The acquired knowledge can be used to select the likely better performing algorithm, means the algorithm that likely delivers the best solution. The question for the best algorithm given a problem instance is not only arising in the context of vehicle routing. It is a general problem, formalised and introduced by Rice in [13] as Algorithm Selection Problem (ASP). The bases for algorithm selection are features characterising the problem instances. For the static version of the VRP several problem features, considering the location of the customer demands, exist, [14]. But features for the dynamic version of the problem are rarely available, [15]. Our work closes this gap by transferring existing location-based TSP and VRP features, which we discuss in Section 1, into dynamic problem features outlined in Section 2. To validate the suitability of our features for algorithm selection we use them for algorithm performance prediction and autonomous algorithm selection supported by simulation. To this end, we first evolve dynamic problem instances from existing static instances with the help of the general purpose metaheuristic optimisation framework SEREIN introduced in [16]. The evolved problem instances are solved with a Greedy and a Re-planning algorithm, which we implemented in the open-source Rich Vehicle Routing Problem Simulator (RVRP Simulator) introduced in [17]. With the generated dynamic instances and the simulation results we are able to investigate our introduced features and whether they are suitable to meaningful distinguish between problem instances. The results of this investigation and the dynamic instance generation are also outlined in Section 2. To validate our problem features in the context of algorithm selection, we implemented a machine learning technique to model the relationship between the problem features and the algorithm performance. The basis for the implemented supervised learning approach is data which were collected with the help of simulation. This model is validated with the grouped cross validation method and additionally evaluated by algorithm performance prediction for unseen problem instances. The model and its evaluation are outlined and discussed in Section 3. Our introduced DVRP features show their validity as base for autonomous algorithms selection. Finally, conclusions are drawn in Section 4.

1 Related Work

This Section introduces the DVRP and the ASP and outlines current research about algorithm selection in the domain of vehicle routing. Additionally common location-based features, used in algorithm selection and performance evaluation studies for the TSP and the VRP, are discussed.

1.1 The Dynamic Vehicle Routing Problem

The DVRP was first introduced in [6] as an extension of the VRP, with a description of a computer-controlled Dial-A-Ride system in Rochester, NY (USA). The VRP, a generalization of the TSP, is defined on a graph $G = (V, E, C)$, where $V = \{v_0, \ldots, v_n\}$ is a set of nodes. The set $E = \{(v_i, v_j) | (v_i, v_j) \in V^2; i \neq j\}$ defines the neighborhood of the nodes V as edge set. $C = (c_{ij})_{(v_i, v_j)} \in E$ defines a cost matrix over E. Nodes are usually customer demands, characterized by a location, which needs to be serviced. The DVRP extends the VRP notion by adding a point in time $(t_i)_{v_i}$ for all v_i, that defines when the customer request appears. In a VRP instance all $(t_i)_{v_i}$ for all v_i would be 0. In a dynamic setting (DVRP) at least one $(t_i)_{v_i}$ would be greater than 0. The TSP, VRP, and DVRP have in common, that the main characteristics of a node $v_i \in V$ is the location.

1.2 The Algorithm Selection Problem

The ASP is formalised and introduced by Rice in [13]. Figure 1 visualises the framework for the ASP proposed in [13]. It aims to predict the best algorithm performance for a problem instance, based on measurable features, [18]. The following four main components are part of the framework.

- the problem space P contains a set of problem instances;
- the feature space F represents a set of measurable features, which can be extracted from all problem

instance in P; the feature extraction process has to be less time-consuming compared to solving the problem instances with algorithms from the algorithm space A;

- the algorithm space A contains a portfolio of algorithms which are able to solve the problem instances from P;
- the performance space Y describes a mapping from algorithm result to a performance metric.

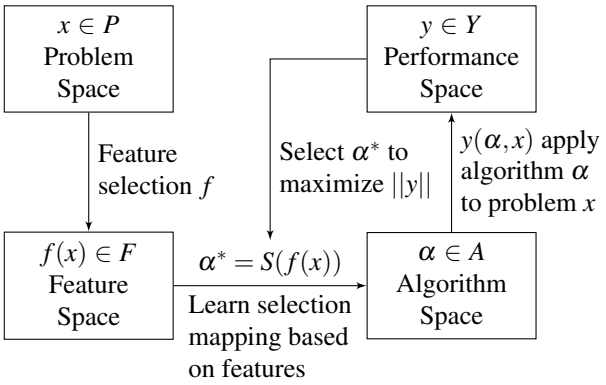

Figure 1: A framework for the Algorithm Selection Problem (ASP) introduced from Rice in [13].

The main challenge of the ASP is to find a selection mapping $S(f(x))$ into the algorithm space A for a problem instance $x \in P$ with feature vector $f(x)$. The mapping has to maximise the performance metric $||y||$ for $y(\alpha,x)$, where α is an algorithm in A. A broad discussion on algorithm selection across a variety of disciplines is given in the survey paper [19].

In the domain of static vehicle routing, [20] investigates the TSP difficulty by learning from evolved instances. The work uses an evolutionary algorithm to evolve TSP instances which are intentionally easy or hard for algorithms based on the Lin-Kernighan heuristic method [21]. Features are derived from the problem instances and the impact of these features for the difficulty of the algorithms is investigated. [22] investigates the success of 2-opt based local search algorithms for solving the TSP and shows important features that make problem instances hard or easy for 2-opt approaches. [23] studies algorithm selection for the Traveling Thief Problem (TTP) based on TSP features. The TTP is a combination of the TSP and the Knapsack Problem (KP). Earlier work focuses on the impact of problem features for the problem difficulty in general. For example, [24] demonstrates that the variance of the distance matrix correlates with the TSP difficulty for exact solving approaches. [25] shows that this is also true for heuristic solving approaches.

1.3 Problem Features

The algorithm selection and performance evaluation studies in the domain of static vehicle routing are usually based on features involving the location of the customer demands. Beside the number of customer demands, [14] classifies the 47 available VRP features, used in their study, into the following eight groups.

- *Distance Features* summarise features which involve the costs of the customer demands. The cost of a demand is the distance from the location of the customer to the depot. Considered are the lowest, highest, mean and median costs of all customer demands. Additionally statistics about the distance matrix, like distinct distances, or the standard deviation of the matrix are categorised in this group.
- *Mode Features* group all features concerning the mode of the customer demand cost distribution, [26]. For all customer the costs to all other customers are calculated. The mode is the cost value which appears most often. Additionally the frequency, quantity and mean of the mode values are considered. [14] additionally groups the number of modes of the customer demand cost distribution introduced in [27] in this category.
- *Cluster Features* bundle features based on statistically information extracted from the results of a clustering algorithm, like the number of clusters or the mean distance of the customer demands to the cluster centroid.
- *Nearest Neighbour Distance Features* group features derived from the calculated nearest-neighbour distances for each customer demand, like the minimum, maximum, mean, or median.
- *Centroid Features* summarise all features calculated dependent on the centroid of all customer demands, including the coordinates of the centroid, minimum, maximum, mean and median of the distances between customer demands and centroid.
- *MST Features* bundle features based on characteristics of the calculated Minimum Spanning Tree

(MST). Considered are the minimum, mean, median, maximum and standard deviation of the depth and distance of the MST.

- *Angle Features* bundle features based on the calculated angle between the two nearest neighbour customer demands, like the minimum, mean, median, maximum and standard deviation of all calculated angles.

- *Convex Hull Features* group the features concerning the convex hull from the problem instance, like the area of the hull and the fraction of customer demands which are part of the hull.

Problem features for the DVRP are rarely described in the literature and are usually discussed independently from algorithm selection. [28] introduced the Degree of Dynamism (DOD), a measure of the dynamism of a dynamic routing problem. The DOD describes the ratio between dynamic customer demands and the sum of dynamic and static customer demands. The DOD is similar to the feature *Number of customer demands* in the context of static vehicle routing. [29] developed a framework based on the DOD. The framework classifies weak, moderate and strong dynamic systems and recommends solution approaches for these classes of dynamic routing problems. [29] also introduced the Effective Degree of Dynamism (EDOD) as extension of the DOD. The EDOD considers the planning horizon T for the calculation of the measure of the dynamism of a routing problem. [29] also studies the relation between DOD, EDOD and routing costs for the Partially Dynamic Traveling Repairman Problem (PDTRP). In [15] we introduced the Location-based Degree of Dynamism (LDOD) capturing the location of the dynamic customer request. We show that there is a positive correlation between our proposed LDOD and the resulting DVRP solution quality (performance) for a Greedy and a Re-planning algorithm. The LDOD is the only feature that focuses on the location of the dynamic customer requests. We categorise the LDOD in the group of *Distance Features*. The LDOD is the only DVRP instance feature that is based on the fundamental characteristic location of a node (customer request). A fundamental problem characteristic is an inherent property, i.e., it is applicable regardless of the problem variation, compare [15].

2 New Features for the Dynamic Vehicle Routing Problem

In this Section DVRP features derived from static VRP location-based features, which are building the Feature Space F, are presented and evaluated. The generation of the Problem Space P, Performance Space Y and Algorithm Space A are also discussed within this Section.

2.1 Feature Space

Since problem features for the DVRP are rarely described in the literature, compare Section 1.3, we had to introduce new problem features. The TSP, the VRP and the DVRP are all defined on a graph G containing a set of nodes $V = \{v_0, \ldots, v_n\}$, compare Subsection 1.1. The main characteristic of a node v_i is its location. Based on this consideration, we are able to base our location-based features for the DVRP on location-based features introduced for the TSP. In general there are two main groups of features we consider to characterise a DVRP problem instance. In the first group we classify existing TSP features which we applied to the dynamic customer demands only. Here we consider *Angle Features*, *MST Features*, *Nearest Neighbour Distance Features*, *Distance Features*, and *Cluster Features*. Features from the second group try to capture the dynamism of the problem instance, by considering both, dynamic and static customer demands. Known examples for the second group are the DOD ([28]) and the LDOD ([15]). For our research we transferred *Angle Features*, and *Nearest Neighbour Distance Features* discussed in Section 2 into features considering dynamic and static demands. Therefore we calculated the distances and angles between dynamic and nearest neighbour static customer demands, compare Figure 2. Since a route to a dynamic customer demand will likely not start at the nearest neighbour static customer, we considered neighbourhoods with size of 2, 3, and 5. For all neighbourhoods we determined features considering the sum, mean, median and maximum value. For example for the neighbourhood of 2 we calculated the distances a and b, compare Figure 2. For the distances we calculated the values $sum(a,b)$, $mean(a,b)$, $median(a,b)$, and $max(a,b)$ for all dynamic customer demands. The resulting features are the sum, maximum, mean, median, standard deviation, and variance coefficient of these calculated values.

We also considered *Centroid Features* and transferred them into features considering dynamic and

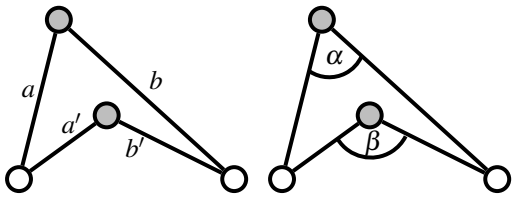

Figure 2: Distances and angles between dynamic customer demands (grey coloured circles) and nearest neighbour static customer demands (white coloured circles).

static customer demands. To this end, we calculated the centroid of the instance including both dynamic and static customer demands. We determined the average and the sum distance from dynamic and the average distance from all static customer demands to the centroid. The resulting feature is the ratio between these two distances. The last feature we introduced is considering the ratio between the surface areas spanned by all dynamic and all static customer demands. Overall we determined 184 different DVRP features.

2.2 Problem Space

To investigate the suitability of our introduced features for algorithm selection we had to create a problem space P and a corresponding performance space Y with algorithms $\alpha \in A$, compare Figure 1. We investigated the suitability of our introduced features with the most basic version a DVRP. We are not considering any capacity constraint or any restrictions regarding the customer requests like time windows or service times. We are focusing on problem instances with one depot and one vehicle only. To expand the problem space as wide as possible, we applied the general purpose metaheuristic optimisation framework SEREIN introduced in [16] to derive dynamic from static VRP instances. Figure 3 visualises the four static VRP instances from [30], which we considered for our research. They have either uniformly distributed (CMT01, CMT04) or clustered (CMT11, CMT12) customer requests.

By deriving dynamic from static problem instances three decisions have to be made. First, how many customer requests shall be dynamic? Second, which requests exactly shall be dynamic? Third, when the customer demand appears during the time horizon? The number of dynamic customer demands is determined by the DOD. In our research the following $DOD \in \{0.1, 0.2, \ldots, 0.9\}$ are considered. The second

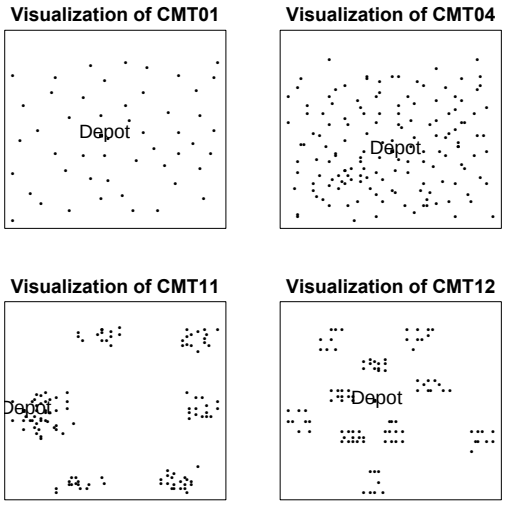

Figure 3: Subset of static VRP instances introduced in [30], considering equal distributed and clustered customer demands (black dots).

decision is made by SEREIN. It selects the determined number of dynamic requests out of all available request. The genetic algorithm evaluates the performance (solution costs) of the created instance with the help of the RVRP Simulator introduced in [17]. The RVRP Simulator provides a set of standard algorithms to handle the dynamic requests. For the problem instance generation we choose a simple Greedy algorithm provided in the simulator. For a more detailed description of the implemented algorithm see the following Section. The solution costs are used to navigate through the search space. The main task of SEREIN is to create dynamic problem instances where the Greedy algorithm performs very good (low solution costs, easy problem instances for the Greedy algorithm) and also instances where the algorithm performs very bad (high solution costs, hard problem instances for the Greedy algorithm). The third decision, when the dynamic requests appear, is not made by the genetic algorithm. Currently, we intend to minimise the influence of the time t_i when request i appears. It is obvious that requests appearing very late are probably producing higher routing costs. For example, if all dynamic requests appear when the vehicle finished its initial route planned for the static customer requests, the employed algorithm cannot integrate the dynamic requests into the existing route. Every time a dynamic request appears, the vehicle has to start from the depot to service the customer, which results in high routing costs. To prevent this interdependency, we partly

follow the approach introduced in [15] and determined the time t_i for a dynamic request i as evenly distributed between 0 and the time horizon T. So, for all created points in time t_i for all dynamic requests n the following rule is valid: $t_{i+1} - t_i = d_i; d_i = d_{i+1} \forall i$, where $t_{n+1} = T$. T is determined by solving the static instance with the Jsprit framework [31]. To prevent the interdependency between the location of request i and the time t_i, we executed the simulation to evaluate the dynamic problem instance several times with different assignments between t_i and i. Additionally, we simulated the instances in a reversed routing order. For example, if a routing starts with servicing the static customer A, B, and ends at static customer C. Additional simulations in reversed order where the routing starts at customer C and ends with servicing customer A are performed. To generate one dynamic problem instance using SEREIN, we had to determine 6 sample routing costs with the help of simulation. The average of these routing costs is used by SEREIN to unfold the search space, identifying instances which are easy as well as instances which are hard to solve for the Greedy algorithm. SEREIN created 2,000 dynamic problem instances for each combination of $DOD \in \{0.1, 0.2, \ldots, 0.9\}$ and problem instance shown in Figure 3. This results in 72,000 dynamic problem instances which got evaluated with 432,000 simulations, only to create the problem space Y.

2.3 Algorithm and the Performance Space

For the Greedy algorithm, we already evaluated the Performance Space Y while creating the Problem Space P. The Greedy algorithm inserts a dynamic request, appearing during problem simulation, into the current planned route for the vehicle. We determined the initial route with the help of the Jsprit framework [31]. The Greedy algorithm evaluates all possible insertion points and chooses the one that produces the best solution costs. A detailed description of the algorithm can be found in [15]. Since we need at least two algorithms to evaluate our introduced DVRP features. We implemented a replanning approach, where every new dynamic request triggers a complete replanning of the tour. So both considered solution approaches are following very contrary ideas. The Re-planning algorithm, that we implemented uses the R package *tspmeta* from [14]. The implementation is based on a 2-opt optimisation algorithm, which was one of the first successful algorithms to solve larger TSP instances and which is still widely used in practice [14]. Due to the stochastic characteristics of the 2-opt approach, we had to simulate each dynamic problem instance (72,000) several times. Overall, we used additional 432,000 simulations (also considering simulations in reverse customer request order) to determine the Performance Space Y for the Re-planning algorithm.

2.4 Evaluation of the new DVRP Features

We separate the generated problem instances into a group of instances which are easy (low solution costs) and into a group of instances which are hard (high solution costs) to solve for each algorithm. We calculate the feature vectors for all instances and evaluate all possible feature combinations and their predictive power to differentiate between hard and easy instances. In this Section we present an exemplary excerpt from our evaluations.

Figure 4 shows the relation between feature *angle_mean* and *mst_dists_mean* from the first group of features (features considering dynamic customer demands only). The graphs in the first row show results derived from the problem instance CMT04. The suitability of the features to distinguish between easy and hard instances is visible by optically separable point clouds for all DOD's. With an increasing DOD, the point clouds are harder to separate from each other. The graphs in the second row consider all instances (compare Figure 3). Here, we observe more point clouds which are harder to separate visually. The borders between the clouds get more indistinct with a decreasing DOD. Similar patterns occurred for other feature combinations from the first group. The evaluation of the features from the second group (features considering static and dynamic customer demands) show similar characteristics, compare Figure 5. But in general there are better and less visually separable point clouds even for higher DOD's. The combination of features of both groups show its suitability to distinguished between hard and easy problem instance too, compare Figure 6.

In general the feature evaluation shows, that the introduced features are suitable to distinguish between easy and hard problem instances for both applied algorithms. Note, that an important feature to distinguish between instances is the DOD, only implicitly shown in the graphs.

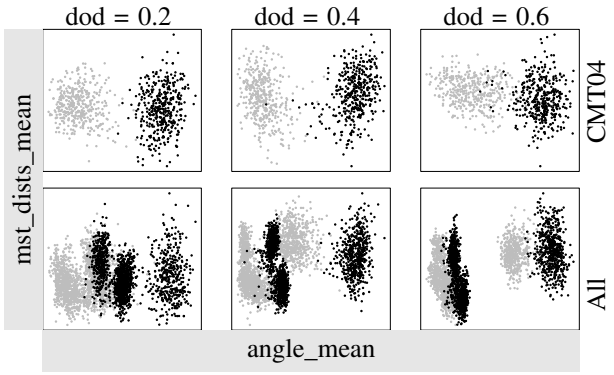

Figure 4: Evaluation of feature combination *angle_mean* and *mst_dists_mean* for Re-planning algorithm.

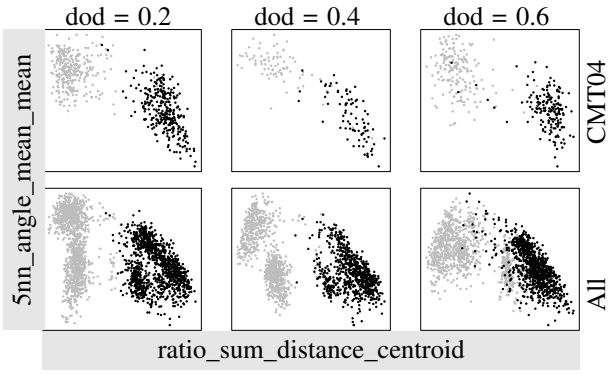

Figure 5: Evaluation of feature combination *ratio_sum_distance_centroid* and *5nn_angle_mean_mean* for Re-planning algorithm.

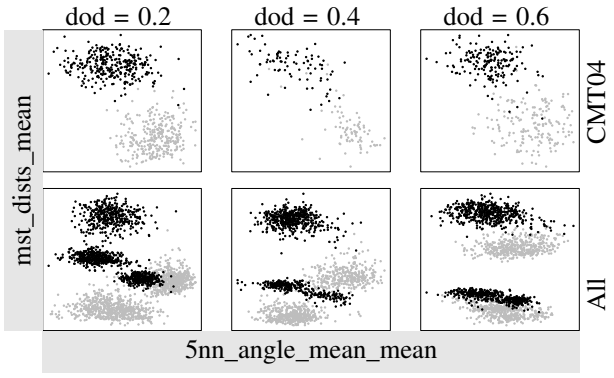

Figure 6: Evaluation of feature combination *mst_dists_mean* and *5nn_angle_mean_mean* for Re-planning algorithm.

3 Algorithm Selection

To test our problem features in the context of Algorithm Selection, we implemented a regression machine learning technique to model the relationship between the problem features and the algorithm performance. We used Artificial Neural Network (ANN) which map unlabeled input to a label (output) using internal data structures. We standardise all values of the feature vectors using the standard scalar method implemented in [32]. These standardised feature vectors are the input for the model (balanced regrading instances and DOD). The output for the regression model is the standardised routing costs for the instances. For each algorithm (Greedy, Re-planning) we constructed one regression model. Due to its popularity we used Tensorflow from Google Brain introduced in [33] for the implementation. Both ANNs have 4 hidden layers with 60, 18, 20, and 22 neurons. The input layer has 184 neurons and the output layer has 1. All neurons are using the activation function RELU, [34]. The structure of the model was setup using systematic trial and error, a common approach to approximate the optimal number of hidden layers and nodes, [35]. All neurons of each layer are fully connected to all neurons of the following layer. We applied the optimisation function *Adam* introduced in [36] to determine the weights of the edges between the neurons of the ANNs. Beside testing our approach with unseen problem instance, we applied the grouped cross validation method to validate our trained ANNs. The method divides the available data into k groups and constructs k different ANNs using $k-1$ data groups. The k^{th} group is used for testing the constructed ANN, [37]. The error is determined using the mean of testing set errors of the groups. The grouped cross validation method is currently considered as the gold standard for testing ANNs. Note, that we had to ensure that the combination of instance and DOD are evenly distributed within all created groups. For our validation we defined 5 data groups. The results of the validation of our constructed ANNs are shown in Table 3. The applied metric for the regression model is the mean squared error (MSE), see for example, [38]. The results in Table 3 show that our features are very suitable for performance prediction. The average error for both ANNs is less than half a percent for all cross validation groups. The models are not over-fitted and precisely predict the algorithm performance, even for unseen dynamic instances generated from the static instances shown in Figure 3.

Algorithm	Mean square error (+/- Standard deviation)
Greedy	0.006775 (+/- 0.000126)
Re-planning	0.003549 (+/- 0.000105)

Table 1: Grouped cross validation results (5 groups) of our constructed ANNs.

To test our regression models with completely unseen problem instances, we created dynamic instances from a subset of VRP instances introduced in [39] and [40], see Figure 7. We created 500 random instances for each instance for each $DOD \in \{0.3, 0.4, \ldots, 0.8\}$ and determined the routing costs with the Greedy and the Re-planning algorithm. The sum of the routing costs for each algorithm, $R = \sum_{i=0}^{n} r_{algorithm.i}$, is shown in Table 3 (column Greedy and Re-planning). Based on the known routing costs for the Greedy and for the Re-planning algorithm, we have been able to determine the correct algorithm selection for each problem instance. The correct selected algorithm is the one which produces the lower costs. The sum of the routing costs, based on a correct algorithm selection, $Correct = \sum_{i=0}^{n} min(r_{greedy.i}, r_{replan.i})$, is also shown in Table 3 (column Correct). Additionally Table 3 shows the sum of the routing costs based on our algorithm selection with our regression (column Regression) models. In general the results, shown in Table 3, illustrate that we are able to successfully select algorithms for unseen problem instances with our approach. Our algorithm selection approach performs better, compared to exclusive use of either the Greedy or the Re-planning algorithm in most cases. However, for the dynamic problem instances derived from problem instance A-n60-k9, our algorithm selection leads to worse results in comparison to the pure Re-Planning algorithm.

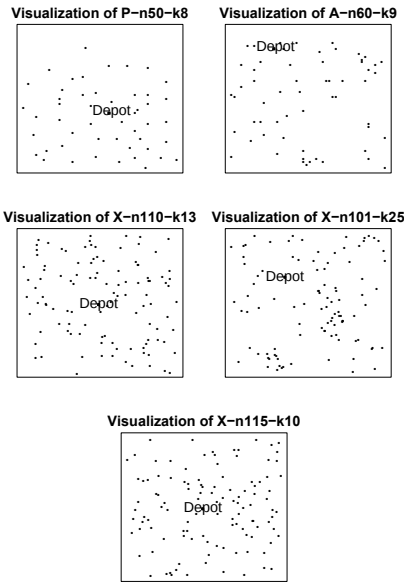

Figure 7: Subset of VRP instances introduced in [39] (P-n50-k8, and A-n60-k9) and [40] (X-n110-k13, X-n101-k25, and X-n115-k10). The dots represent the customer requests.

4 Conclusion and Outlook

Our results show that our approach is able to reliably autonomously select the better performing algorithm for various known and partly also for unseen DVRP instances. Based on simulation results we have been able to derive characteristic DVRP from static problem instances. We developed and calculated features for these instances. Our introduced features in combination with the simulation results are the bases for our developed regression models which we trained to predict algorithm performance. The prediction of algorithm performance is the base for our successful autonomous algorithm selection.

For our performance prediction models we are currently considering only 4 different structured problem instances, compare Figure 3. The trained regression models are quite fitted to these instances which seems to be the reason for the difficulties with performance prediction for the unseen problem instances A-n60-k9, compare Table 3. The idea is to build a stronger basis for our performance prediction by considering more different kinds of problem instances. We plan to include the very geometrical structured VRP instances introduced in [41] in our research. We also have to evaluate the possibility to create own artificial instances.

Currently we only consider two different algorithms which perform similar for different problem instances, compare Section 3. Future work will be the training and testing of regression models based on simulation results of different DVRP algorithms. We will implement an DVRP algorithm portfolio with existing algorithms. We are currently developing a new DVRP solving approach based on the results from [15]. Beside considering more instances and more algorithms, the question occurs which evolved instances are good for training

3000 Instances based on	Greedy	Re-planning	Correct	Regression
X-n110-k13	31,664,462	31,836,229	*31,405,740*	**31,413,135**
X-n101-k25	34,794,388	34,725,471	*34,389,425*	**34,455,935**
X-n115-k10	39,754,906	39870583	*39402544*	**39,429,201**
P-n50-k8	534,005	534,690	*528,157*	**533,961**
A-n60-k9	2,787,325	**2,781,479**	*2,752,555*	2,783,146

Table 2: Validation results of our constructed ANNs for unseen instances.

and testing purposes. Currently we are not verifying if an evolved instances is a good representation for the performance class. In future work, we have to determine a metric which provides information about how good a certain instance represents a certain algorithm performance.

References

[1] Dantzig GB, Ramser JH. The Truck Dispatching Problem. *Management science*. 1959;6(1):80–91.

[2] Flood MM. The traveling-salesman problem. *Operations Research*. 1956;4(1):61–75.

[3] Lahyani R, Khemakhem M, Semet F. Rich Vehicle Routing Problems: From a Taxonomy to a Definition. *European Journal of Operational Research*. 2015; 241(1):1–14.

[4] Pillac V, Gendreau M, Guéret C, Medaglia AL. A Review of Dynamic Vehicle Routing Problems. *European Journal of Operational Research*. 2013; 225(1):1–11.

[5] Psaraftis HN, Wen M, Kontovas CA. Dynamic vehicle routing problems: Three decades and counting. *Networks*. 2016;67(1):3–31.

[6] Wilson NH, Colvin NJ. *Computer Control of the Rochester Dial-a-ride System*. 77-22. Cambridge, Massachusetts. 1977.

[7] Ritzinger U, Puchinger J, Hartl RF. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*. 2016; 54(1):215–231.

[8] Clarke G, Wright JW. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations research*. 1964;12(4):568–581.

[9] Juan AA, Faulin J, Jorba J, Riera D, Masip D, Barrios B. On the use of Monte Carlo Simulation, Cache and Splitting Techniques to Improve the Clarke and Wright Savings Heuristics. *Journal of the Operational Research Society*. 2011;62(6):1085–1097.

[10] Juan AA, Faulin J, Grasman SE, Rabe M, Figueira G. A Review of Simheuristics: Extending Metaheuristics to Deal with Stochastic Combinatorial Optimization Problems. *Operations Research Perspectives*. 2015; 2:62–72.

[11] Gruler A, Klüter A, Rabe M, Juan AA. A Simulation-Optimization Approach for the Two-Echelon Location Routing Problem Arising in the Creation of Urban Consolidation Centres. In: *Simulation in Produktion und Logistik 2017*, edited by Wenzel S, Peter T. Kassel: kassel university press. 2017; pp. 129–138.

[12] Wolpert DH, Macready WG. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*. 1997;1(1):67–82.

[13] Rice JR. The Algorithm Selection Problem. *Advances in Computers*. 1976;15:65–118.

[14] Mersmann O, Bischl B, Trautmann H, Wagner M, Bossek J, Neumann F. A Novel Feature-based Approach to Characterize Algorithm Performance for the Traveling Salesperson Problem. *Annals of Mathematics and Artificial Intelligence*. 2013;69(2):151–182.

[15] Mayer T, Uhlig T, Rose O. A Location Model for Dynamic Vehicle Routing Problems. In: *Simulation in Produktion und Logistik 2017*, edited by Wenzel S, Peter T. Kassel: kassel university press. 2017; pp. 149–158.

[16] Uhlig T. *Self-Replicating Individuals*. München: Verlag Dr. Hut. 2015.

[17] Mayer T, Uhlig T, Rose O. An open-source discrete event simulator for rich vehicle routing problems. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, edited by Rossetti R, Wolf D. New York, USA: IEEE. 2016; pp. 1305–1310.

[18] Smith-Miles K, Lopes L. Measuring Instance Difficulty for Combinatorial Optimization Problems. *Computers & Operations Research*. 2012;39(5):875–889.

[19] Smith-Miles KA. Cross-disciplinary Perspectives on Meta-learning for Algorithm Selection. *ACM Computing Surveys (CSUR)*. 2009;41(1):6.

[20] Smith-Miles K, van Hemert J, Lim XY. Understanding TSP Difficulty by Learning from Evolved Instances. In: *Proceedings of the 4th International Conference on Learning and Intelligent Optimization*, edited by Blum C, Battiti R. Berlin, Heidelberg: Springer-Verlag. 2010; pp. 266–280.

[21] Lin S, Kernighan BW. An Effective Heuristic Algorithm for the Traveling-salesman Problem. *Operations Research*. 1973;21(2):498–516.

[22] Mersmann O, Bischl B, Bossek J, Trautmann H, Wagner M, Neumann F. Local Search and the Traveling Salesman Problem: A Feature-Based Characterization of Problem Hardness. In: *Learning and Intelligent Optimization*, edited by Hamadi Y, Schoenauer M, pp. 115–129. Berlin, Heidelberg: Springer Berlin Heidelberg. 2012;.

[23] Wagner M, Lindauer M, Misir M, Nallaperuma S, Hutter F. A Case Study of Algorithm Selection for the Traveling Thief Problem. *Journal of Heuristics*. 2018; 24(3):295–320.

[24] Cheeseman P, Kanefsky B, Taylor WM. Where the Really Hard Problems Are. In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, edited by Mylopoulos J, Reiter R. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 1991; pp. 331–337.

[25] Ridge E, Kudenko D. An Analysis of Problem Difficulty for a Class of Optimisation Heuristics. In: *Proceedings of the 7th European Conference on Evolutionary Computation in Combinatorial Optimization*, edited by Cotta C, Van Hemert J. Berlin, Heidelberg: Springer-Verlag. 2007; pp. 198–209.

[26] Kanda J, Carvalho A, Hruschka E, Soares C. Selection of algorithms to solve traveling salesman problems using meta-learning 1. *International Journal of Hybrid Intelligent Systems*. 2011;8(3):117–128.

[27] Mersmann O, Bischl B, Trautmann H, Preuss M, Weihs C, Rudolph G. Exploratory landscape analysis. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011; pp. 829–836.

[28] Lund K, Madsen OB, Rygaard JM. Vehicle Routing with Varying Degree of Dynamism. 1996; (IMM-REP-1996-1).

[29] Larsen A. The Dynamic Vehicle Routing Problem. Ph.D. thesis, Institute of Mathematical Modelling, Technical University of Denmark. 2000.

[30] Christofides N. The Vehicle Routing Problem. *Revue française d'automatique, informatique, recherche opérationnelle Recherche opérationnelle*. 1976; 10(V1):55–70.

[31] Schröder. Jsprit. `https://github.com/graphhopper/jsprit`. 2018. Accessed July 2nd, 2018.

[32] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*. 2011; 12(11):2825–2830.

[33] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X. TensorFlow: A System for Large-scale Machine Learning. In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16. Berkeley, CA, USA: USENIX Association. 2016; pp. 265–283.

[34] Nair V, Hinton GE. Rectified Linear Units Improve Restricted Boltzmann Machines. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML)*, edited by Fürnkranz J, Joachims T. USA: Omnipress. 2010; pp. 807–814.

[35] Basheer IA, Hajmeer M. Artificial Neural Nnetworks: Fundamentals, Computing, Design, and Application. *Journal of microbiological methods*. 2000;43(1):3–31.

[36] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 2015;.

[37] Twomey JM, Smith AE. *Validation and Verification*. New York, NY, USA: American Society of Civil Engineers. 1997.

[38] Lehmann EL, Casella G. *Theory of Point Estimation*. 175 Fifth Avenue, New York: Springer Science & Business Media, 2nd ed. 2006.

[39] Christiansen CH, Lysgaard J. A Branch-and-price Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands. *Operations Research Letters*. 2007;35(6):773–781.

[40] Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A. New Benchmark Instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*. 2017; 257(3):845–858.

[41] Golden BL, Wasil EA, Kelly JP, Chao IM. The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. In: *Fleet Management and Logistics*, edited by Crainic TG, Laporte G, pp. 33–56. Boston, MA: Springer US. 1998;.