Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

SIMULATION-BASED AUTONOMOUS ALGORITHM SELECTION FOR DYNAMIC VEHICLE ROUTING PROBLEMS WITH THE HELP OF SUPERVISED LEARNING METHODS

Thomas Mayer
Tobias Uhlig
Oliver Rose

Department of Computer Science
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39
Neubiberg, 85579, GERMANY

ABSTRACT

Multi-constrained Vehicle Routing Problems are gaining steadily in importance. Especially, the dynamic version of the problem has become more emphasis due to modern service requirements, such as short-term or express delivery. With a growing number of dedicated solution approaches for these problems, we investigate a simulation-based supervised learning approach to determine the suitability of a particular algorithm from a set of algorithms for a given dynamic problem instance based on a variety of its characteristics. This decision is known as the Algorithm Selection Problem. We explore the performance space for Greedy and Re-planning algorithms for different dynamic problem instances by simulation and an evolutionary algorithm. For the algorithm selection we test several problem features in combination with two supervised machine learning techniques. The applicability of our approach is demonstrated in a use case for autonomous algorithm selection for Dynamic Vehicle Routing Problem instances.

1 INTRODUCTION AND RELATED WORK

Our modern way of life depends more and more on managing and solving complex Vehicle Routing Problems (VRPs). For example, our groceries are delivered fast and fresh to our doorstep. We order different goods for daily use over the Internet, we share cars and use on-demand transportation services, and we expect everything to be in stock in the nearest supermarket. The fundamental Vehicle Routing Problem (VRP) was introduced in Dantzig and Ramser (1959). Essentially, it encompasses the planning of routes for vehicles to satisfy customer requests starting and ending at a depot. Today, they are a large variety of VRP classes. A current survey and taxonomy is, for example, Lahyani et al. (2015). For static VRP instances, all relevant information is available prior to solving the problem. This a priori information can be either deterministic or stochastic if uncertainties like traffic are considered (Pillac et al. 2013). In real-world applications, however, the evolution of the problem information over time is a major problem characteristic (Psaraftis 1980). Partly, problem information becomes only available during route execution, for example, due to additional customer demands. Problem information which changes over the time is classified as dynamic (Pillac et al. 2013). Therefore, the problem class with changing information is referred to as Dynamic VRP (DVRP). The DVRP was first introduced in Wilson and Colvin (1977) as an extension of the VRP, with a description of a computer-controlled Dial-A-Ride system in Rochester, NY (USA). A recent overview on dynamic and stochastic vehicle routing is Ritzinger et al. (2016). Simulation is commonly used to handle the dynamic and stochastic aspects of vehicle routing. For example, Juan et al. (2011) improve the heuristic from Clarke and Wright (1964) by using Monte Carlo simulation. Juan et al.

(2015) provide an extensive review of approaches using simulation to solve combinatorial optimization problems. These authors introduce the concept of Simheuristics, a methodology that integrates simulation into the solution finding process for combinatorial optimization problems. This methodology is also used in Gruler et al. (2017) to solve the Two-Echelon Location Routing Problem, a combination of the Capacitated Location Routing Problem (CLRP) and a VRP. Psaraftis et al. (2016) provide a broad overview on solution methods mainly addressing the DVRP. Since the late seventies, considerable research has focused on the dynamic and stochastic aspects in vehicle routing. Pillac et al. (2013) and Psaraftis et al. (2016) describe an explosion of the number of related papers after the year 2000. Most of these papers focus on algorithm and solution development. With regard to the No Free Lunch Theorem introduced in Wolpert and Macready (1997), we know that there is no strictly superior heuristic or parameterization for a solving framework that outperforms all other heuristics or all other parameterizations for all problem instances. Due to the intense research in this area and the increased number of algorithms and solution approaches for different variations of the VRP, 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 likely better performing algorithm? In addition, there will be difficulties in comparing published results and employed approaches (Ritzinger et al. 2016).

The question about the best algorithm for a given problem instance is not only arising in the context of vehicle routing. It is a general problem, formalized and introduced in Rice (1976) as the Algorithm Selection Problem (ASP). Figure 1 visualizes a framework for the ASP proposed in Rice (1976). The framework aims to support the prediction of the best algorithm performance for a problem instance based on measurable features (Smith-Miles and Lopes 2012). The framework visualized in Figure 1 describes the following four main components:

- 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 instances 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 that are able to solve the problem instances from P;
- the performance space Y describes a mapping from algorithm results to a performance metric.

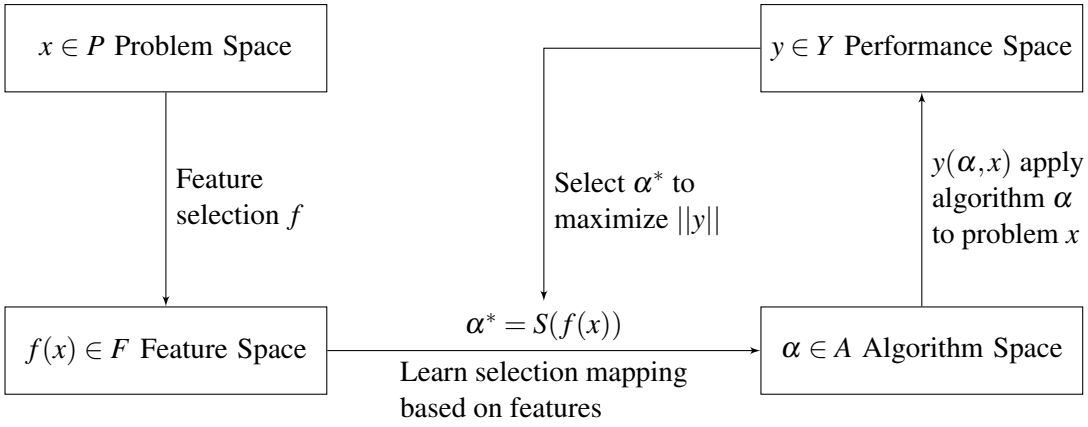

Figure 1: A framework for the Algorithm Selection Problem (ASP) introduced in Rice (1976).

The main challenge of the ASP is to find a selection mapping $S(f(x))$ for a problem instance $x \in P$ with feature vector $f(x)$ into the algorithm space A. The mapping has to maximize the performance metric $||y||$ for $y(\alpha,x)$, where α is an algorithm in A. In general, there is a lot of research in the area of performance

prediction for algorithms that can be related to the ASP. A broad discussion about algorithm selection across a variety of disciplines is given in the survey paper Smith-Miles (2009). In the domain of vehicle routing, Smith-Miles et al. (2010) investigate the Traveling Salesman Problem (TSP) difficulty by learning from a large number of instances. The work uses an evolutionary algorithm to evolve TSP instances that are intentionally easy or hard for algorithms based on the Lin-Kernighan heuristic method (Lin and Kernighan 1973). Features are derived from the problem instances and the impact of these features for the difficulty of the algorithms is investigated. Mersmann et al. (2012) investigate the success of 2-opt-based local search algorithms for solving the TSP and show important features that make problem instances hard or easy for 2-opt approaches. Wagner et al. (2018) study 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 to the problem difficulty in general. For example, Cheeseman et al. (1991) demonstrate that the variance of the distance matrix correlates with the TSP difficulty for exact solution approaches. Ridge and Kudenko (2007) show that this is also true for heuristic solution approaches.

Early work in the domain of dynamic vehicle routing focuses on intrinsic characteristics of problem instances independently from algorithm selection. Lund et al. (1996) introduced the Degree of Dynamism (DOD), a measure of the dynamism of a routing problem. The DOD describes the ratio between dynamic customer requests and the sum of dynamic and static customer requests. The DOD can be interpreted as a very general feature $f(x) \in F$ for a dynamic routing problem $x \in P$ in the context of the ASP. Larsen (2000) 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. Larsen (2000) also introduced the Effective Degree of Dynamism (EDOD) as extension to the DOD. The EDOD considers the planning horizon T for the calculation of the measure of the dynamism of a routing problem. Larsen (2000) also studies the relation between DOD, EDOD, and routing costs for the Partially Dynamic Traveling Repairman Problem (PDTRP). In Mayer et al. (2017) 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 the proposed LDOD and the resulting DVRP solution quality for Greedy and Re-planning algorithms. In the context of the ASP, the LDOD is also a very predictive feature $f(x)$ of the feature space F for a DVRP.

An evolutionary hyper-heuristic which is able to evolve and generate sophisticated sequences of existing low-level heuristics to solve a DVRP is introduced in Garrido and Riff (2010). A hyper-heuristic uses a higher-level heuristic to select among simpler heuristic search algorithms (Burke et al. 2003) and can be discussed within the ASP Framework shown in Figure 1. In the context of a hyper-heuristic, features F are not intrinsic characteristics of the problem instances like the DOD or the LDOD. The performance of the simpler heuristics can be seen as features F (Smith-Miles and Lopes 2012).

Our research focuses on the investigation, whether the discussed intrinsic characteristics and new ones can be used to distinguish between problem instances, and if it is possible to autonomously select algorithms based on the measured features between instances. Therefore, we take the existing DVRP instance features from the literature and use them in combination with new problem features for algorithm performance prediction and autonomous algorithm selection. The basis for the implemented supervised learning approach is data which were collected with the help of simulation. To this end, we first generate dynamic problem instances from existing static instances with the help of the general purpose metaheuristic optimization framework SEREIN (Uhlig 2015). Existing and new problem features that capture the dynamism of the generated instances are applied. The performance space is constructed based on Greedy and Re-planning algorithms, which we implemented in the open-source Rich Vehicle Routing Problem Simulator (RVRP Simulator) introduced in Mayer et al. (2016). The problem and performance space generation is outlined in Section 2. With the help of the results from the simulations, we are able learn the relationship between the problem features and the algorithm performance. Section 3 focuses on the feature space and the comparison of two machine learning techniques that we used to represent the relationship between features and algorithm

performance. Our approach is evaluated by algorithm performance prediction for fresh problem instances, which we outline and discuss in Section 4. Our approach shows its validity for automated algorithms selection for DVRP instances. Conclusions are drawn in Section 5.

2 PROBLEM AND PERFORMANCE SPACE

The DVRP is a very generic problem where different authors consider different variations (Psaraftis et al. 2016). All variations have in common that they are addressed with the help of simulation. The changing problem information over time makes the DVRP well suited for Discrete Event Simulation. Currently, our research focuses on the most basic version of a DVRP. We not consider any capacity constraints or any restrictions regarding the customer requests such as time windows or service times. The focus lies on problem instances with one depot and one vehicle only. Note that instances that are partly dynamic, i.e., $0 < DOD < 1$, are considered only. In general, there is a lack of DVRP instances. Our research shows that out of 50 recent papers discussing solution approaches for the DVRP, $> 60\%$ are evaluating their solutions purely with random problem instances, or instances which are based on real-world data. Another $> 25\%$ are randomizing static problem instances. Only 9% of the considered papers are evaluating their approaches with existing public DVRP instance repositories. Most of these papers are using instances introduced in Kilby et al. (1998) or Bent and Van Hentenryck (2004). The reason for this small percentage is the lack of DVRP instances with various restrictions and constraints. Due to the described shortage of suitable dynamic problem instances, we are following the approach to derive dynamic from static instances. Figure 2 visualizes the four static VRP instances from Christofides (1976) that we considered for our research. They have either equally distributed (CMT01, CMT04) or clustered (CMT11, CMT12) customer requests.

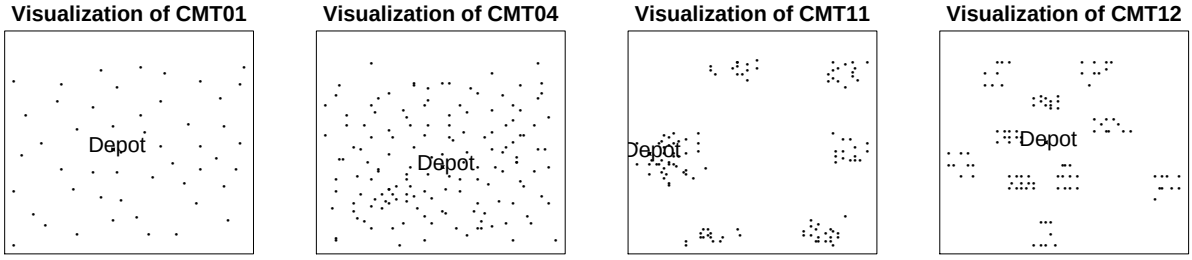

Figure 2: A subset of static VRP instances introduced from Christofides (1976), considering equally distributed (CMT01, CMT04) and clustered (CMT11, CMT12) customer requests which are represented by dots.

To derive dynamic from static DVRP instances, two decisions have to be made. The first decision is which customer requests should be dynamic and the second is when the requests appear during the planning time horizon. The number of dynamic customer requests is determined by the DOD, which, in our case, is smaller than 1. To expand the performance space Y for an algorithm $\alpha \in A$, compare Figure 1, as wide as possible, the decision which customer requests should be dynamic is made by a genetic algorithm. The main task of the genetic algorithm is to find dynamic problem instances where the algorithm performs very well and also instances where the same algorithm performs very badly. For our research, we applied the genetic algorithm implemented in the general purpose metaheuristic optimization framework SEREIN (Uhlig 2015). SEREIN evaluates the created instance by means of the RVRP Simulator introduced in Mayer et al. (2016). The RVRP Simulator provides a set of standard algorithms to handle the dynamic requests. For the evaluation, we choose a simple Greedy algorithm provided by the simulator. For a more detailed description of the implemented algorithm see Mayer et al. (2017). The second decision, when the dynamic requests appear, is not made by the genetic algorithm. Currently, we intend to minimize 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 Mayer et al. (2017) 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 (Schröder 2018). 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 created from SEREIN, we had to determine 6 different 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 and 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 2. This results in 72,000 dynamic problem instances which got evaluated with 432,000 simulations, only to investigate the performance space Y for the Greedy algorithm. Each of the 72,000 problem instances we also solved with a Re-planning algorithm which we implemented for the RVRP Simulator. We did not use the Re-planning algorithm described in Mayer et al. (2017). A new algorithm using the R package *tspmeta* from Mersmann et al. (2013) was implemented. The implementation bases on a 2-opt optimization algorithm, which was one of the first successful algorithms to solve larger TSP instances and which is still widely used in practice (Mersmann et al. 2013). Due to the stochastic characteristics of the Re-planning algorithm, each dynamic problem instance was simulated several times. Overall, we used an additional 432,000 simulations (also considering simulations in reverse customer request order) to collect the results for the Re-planning algorithm. An excerpt of the results of the simulations is shown in Figure 3.

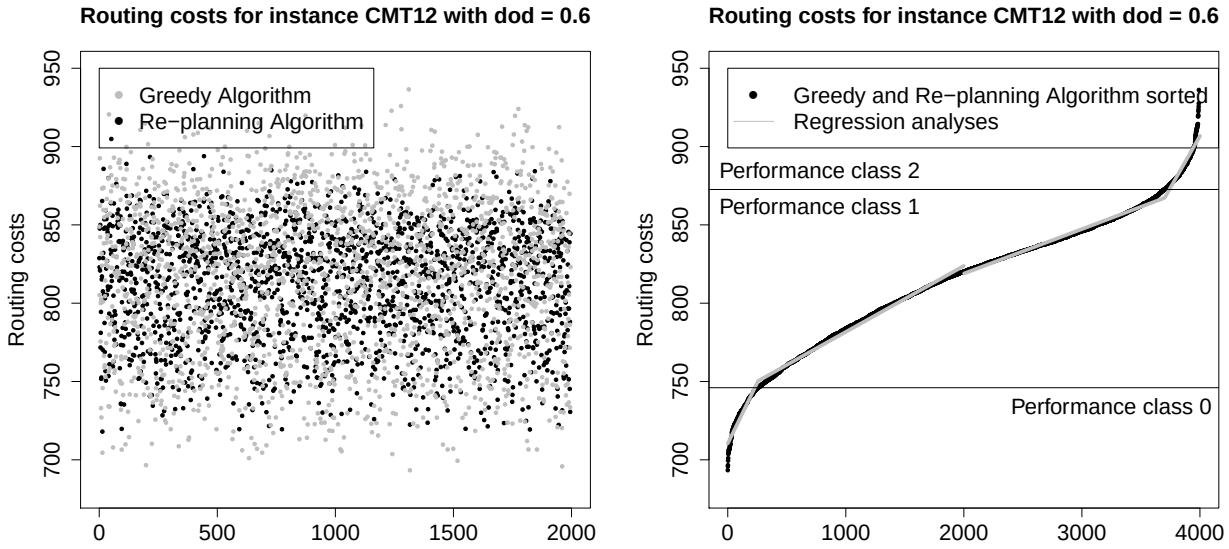

Figure 3: The left diagram shows the routing costs determined with the help of simulation for the Greedy and Re-planning algorithm. All problem instances are derived from CMT12 with a DOD of 0.6 with the help of SEREIN. The right diagram shows the sorted routing costs and determined performance classes.

The left diagram in Figure 3 shows the average routing costs for the DVRP instances with a DOD of 0.6. The instances are derived from the static VRP instance CMT12. We observe that there are instances

hard to solve (high costs) and easy to solve (low costs) for both algorithms. The performance of the Re-planning algorithm is more or less average for all instances. We conclude that there are instances where the Greedy algorithm outperforms the Re-planning algorithm and vice versa. Our research challenge is now to distinguish between these instances based on measurable instance features to make reasonable algorithm selections (Figure 1). Two different machine learning techniques to identify these instances are implemented, a classification and a regression model. Both approaches are realized with Artificial Neural Networks (ANN), see the following Section. A regression analysis estimates the relationships among variables, in our case the relation between the problem instance and the algorithm performance. A classification model assigns categories to given inputs. Therefore, for the classification model, the routing costs and created performance classes had to be evaluated first. For the human eye it is easy to see that instances with routing costs below 750 are somehow special or different from instances with routing costs above 750 due to the density of the visualized results. The same is valid for instances with routing costs around 860. To determine these borders algorithmically, a regression analysis with breakpoint estimation was applied to the sorted routing costs. We used the regression model introduced in Muggeo (2003). With this model, the borders of the performance classes can be estimated very well, see the right diagram in Figure 3. For each created DVRP instance the associated performance class for each algorithm is defined. Three performance classes for each instance of each DOD are considered. In summary, we obtained the performance of each algorithm for the generated instances. From there, a differentiation between these instances based on measurable features is needed. We try to learn which features have an influence and are significant to predict algorithm performance. By understanding which feature combination leads to good algorithm performance, algorithm performance prediction for unseen instances which is the base for automated algorithm selection gets possible. In the following Section, the selected features that are used to distinguish between problem instances are discussed. The following Section will also introduce our implemented machine learning approaches, which were employed to learn the relation between features and algorithm performance.

3 FEATURE SPACE AND SELECTION MAPPING

There are several features for TSP instances available in the literature. For example, Wagner et al. (2018) base their case study on algorithm selection for the TTP on 47 different TSP features. We applied a subset of these features to determine statistics for the dynamic requests, see Table 1. Additionally, we developed new features mainly describing a ratio between the location of the dynamic and the location of all customer requests, also part of Table 1. They are capturing the dynamism of the problem instance and are partially based on existing TSP features.

We calculated all features from Table 1 for all 72,000 created DVRP instances. We standardize all values using the standard scalar method implemented in Pedregosa et al. (2011). These standardized feature vectors are the input for both models (classification and regression). Artificial Neural Networks (ANNs) map an un-labeled input (the standardized feature vector) to a label (output) using internal data structures. The performance classes associated to the instances in a one-hot encoded form are the output for the classification model. The outputs for the regression model are the standardized routing costs of the instances. For each algorithm (Greedy, Re-planning) we constructed a classification and a regression model. In total, we trained and tested four ANNs. Due to its popularity, we used Tensorflow from Google Brain introduced in Abadi et al. (2016) for the implementation. Both ANNs used for classification have 4 hidden layers with 11, 10, 9, and 8 neurons. The input layer has 18 neurons and the output layer has 3. The neurons from the input and hidden layers are using the activation function RELU (Nair and Hinton 2010), the neurons from the output layer use SOFTMAX (Bridle 1990). The structure of the ANNs used for regression is slightly different. Here, we implemented 4 hidden layers with 9, 10, 11, and 12 neurons. The input layer has 30 and the output layer has 1 neuron. All neurons are using the activation function RELU. The structures of the classification and the regression model were set up using systematic trial and error, a common approach to approximate the optimal number of hidden layers and neurons

Table 1: TSP instances features we applied on dynamic customer requests and features describing a ratio between the location of the dynamic and all customer requests.

TSP instances features	Description
$ANGLE_{MEAN}$, $ANGLE_{VAR}$	Statistics of the distribution of the angle between a dynamic customer request and its two next dynamic neighbors
MST_{MEAN}, MST_{VAR}	Construct minimum spanning tree (MST) for dynamic customer requests, then calculate the statistics of the distances
NND_{MEAN}, NND_{VAR}	Distribution of distances of dynamic customer requests to its neighbors.
$DISTANCE_{DISTINCT}$, $DISTANCE_{VAR}$	Computes statistics describing the distribution of pairwise distances between dynamic customer requests.
$CLUSTER_{NUMBER}$	Mean distances from all dynamic requests in a cluster to its centroid.
Ratio Features	**Description**
DOD	The Degree of Dynamism introduced in Lund et al. (1996).
$EDOD$	The Effective Degree of Dynamism introduced in Larsen (2000)
$LDOD$	The Location based Degree of Dynamism (Mayer et al. 2017)
$LDOD_{DEPOT}$	Ratio between the distances of dynamic requests to all requests to the depot.
$AREA$	The ratio between the covered area of dynamic to all requests.
$CENTROID_{MEAN}$, $CENTROID_{SUM}$	The ratio between the mean and the sum of distances of dynamic requests to the centroid to all customer requests to the centroid.

(Basheer and Hajmeer 2000). All neurons of each layer are fully connected to all neurons of the following layer. We applied the optimization function *Adam* introduced in Kingma and Ba (2015) to determine the weights of the edges between the neurons of the ANNs. Especially for classification problems addressed with ANNs, balancing the training data is important (Basheer and Hajmeer 2000). It is recommended that the input data are evenly distributed between the classes (Swingler 1996). It should be prevented that the network is biased towards the over-represented classes. In our case, we had to ensure that not only the performance classes 0, 1, and 2 are evenly distributed, we also had to ensure that all instances with all DODs and all performance classes are evenly represented in the training and test data for the classification models. Therefore, we determined the minimum available datasets with the same instance, DOD, and performance class $min_{i-DOD-pc}$. This minimum defines the maximum of datasets with the same combination of instance, DOD, and performance class we considered for the training and testing of our classification ANNs. The total number of datasets we consider for one algorithm is determined as follows: $number_{total} = number_{instance} * number_{DOD} * number_{classes} * min_{i-DOD-pc}$. For the Greedy algorithm, where $min_{i-DOD-pc}$ is 38, we used $4,104$ out of the $72,000$ available datasets. The reason for the gap is the very over-represented performance class 1, compare Figure 3. For the Re-planning algorithm, $min_{i-DOD-pc}$ is < 20. To avoid working with too little data, we defined a minimum of 20 for $min_{i-DOD-pc}$. For combinations of instances, DOD, and performance classes where the defined minimum is not reached, all available datasets are taken into account. As a consequence, those concerned combinations are under-represented in the ANN trainings and test data. For the regression models, we only had to balance the input data regarding instances and DOD. Beside testing our approach with unseen problem instances, see the following section, 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 (Twomey and Smith 1997). 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. We had to ensure that the combinations of instances, DOD, and performance classes 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 2. The shown value for categorical accuracy

for the classification model can be interpreted as percentage of how often the predicted value matches the true value. The applied metric for the regression model is the mean squared error (MSE), see for example, Lehmann and Casella (2006).

Table 2: Grouped cross validation results of our constructed ANNs. σ is the Standard deviation.

	Classification model	Regression model
Algorithm	Categorical accuracy (+/- σ)	Mean square error (+/- σ)
Greedy	70.84% (+/- 2.97%)	0.079 (+/- 0.098)
Re-planning	85.60% (+/- 4.07%)	0.067 (+/- 0.103)

With a categorical accuracy greater than 70 percent, our classification ANNs perform significantly better than a random classification method. A random method would have an average accuracy of 33% considering the three performance classes to predict. But, nevertheless, in close to 30% or 15% of all cases our classification models predicted the wrong performance class. The main reason is the amount of data for training and testing. Due to the small number of instances with the performance classes 0 and 2, compare Figure 3, the overall data for learning and training is small. The evaluation of the wrongly classified problem instances shows that the wrongly predicted performance classes are usually neighbors of the correct classes. For example, if the ANN predicts the wrong performance class 2, usually the correct class is 1 and not 0. Additionally, these instances are usually close to the artificially defined performance class borders. That means that in general the distinction between instances with good and bad performance is possible based on our selected features. The problems arise at the performance class borders which would have a less negative effect if there were more data for training and testing. For the regression model, which uses all the available data, the MSE is reasonably low, the predicted results differ on average only 8% from the true values. The results show a relatively high standard deviation compared to the mean. That means that for some randomly created groups of data (cross validation) the prediction is very successful and for others only moderately successful. A deeper discussion about the performance and future work is given in Section 5. In the following Section, we will use the classification and the regression models for algorithm selection for unseen problem instances.

4 ALGORITHM SELECTION FOR UNSEEN PROBLEM INSTANCES

To test our classification and regression models with unseen problem instances, we created dynamic instances from a subset of VRP instances introduced in Christiansen and Lysgaard (2007). The employed instances are shown in Figure 4. 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. Each simulation needed to determine the costs for one instances for one algorithm is repeated five times. The average of the simulation results for one instance i is defined as costs ($r_{greedy.i}$, $r_{replan.i}$). The sum of these 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 an algorithm selection with the classification (column Classification) and the regression (column Regression) models. Note, that we cannot make a statement about which algorithm to select if the predicted performance classes from the Greedy and the Re-planning classification models are identical. In this case, a random algorithm is chosen. Due to this limited explanatory power for identically predicted performance classes, we introduced an approach that uses the regression model to select algorithms if the classification models predict the same performance class. The sum of the routing costs based on this mixed approach is also shown in Table 3 (column Mixed).

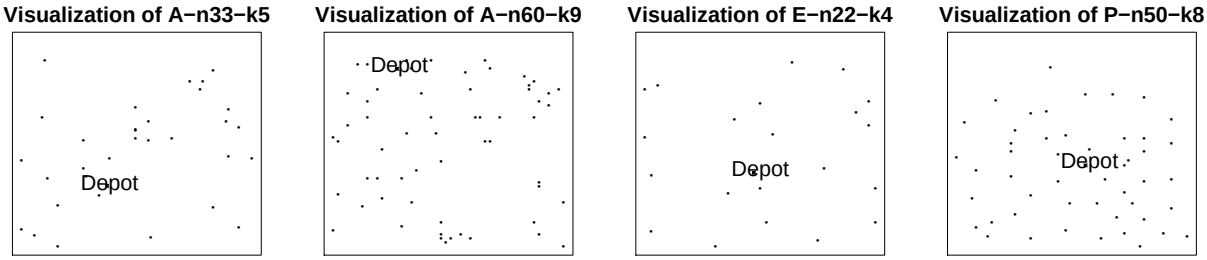

Figure 4: A subset of the stochastic VRP instances introduced in Christiansen and Lysgaard (2007). The dots represent the customer requests.

In general, the results show that we are able to successfully select algorithms for unseen problem instances with our approach. All our created selection methods (Classification, Regression, and Mixed) perform better compared to only taking either the Greedy or the Re-planning algorithm. The values from the correct selection (Table 3 column Correct) show, that both algorithms perform quite similar for all problem instances. The benefit of selecting the better performing algorithm amounts to approximately 1.5% for all randomly created dynamic problem instances based on the instances shown in Figure 4. This makes algorithm selection very difficult, because there are no problem instances where both algorithms perform significantly different. The predicted algorithm performance for both algorithms is likely to be very close to each other and the error probability is therefore high. On average our selection methods make the correct decision in 60% of the cases. Our results show that, in summary, decisions made with our selection methods are correct. In particular, whenever a decision has a significant impact, i.e., instances where different algorithms lead to considerably different results.

Table 3: Validation results of our constructed ANNs for unseen instances.

3,000 instances of	Greedy	Re-planning	Correct	Classification	Regression	Mixed
A-n33-k5	2012196	2001882	*1981276*	2001861	**2001140**	2001506
A-n60-k9	2796131	2781403	*2752825*	2781382	**2781195**	2781284
E-n22-k4	1244571	1241528	*1229514*	1241331	1241393	**1241245**
P-n50-k8	1690522	1690202	*1669857*	1690191	**1688828**	1689087

5 CONCLUSION AND OUTLOOK

Our results show that our approach is able to reliably autonomously select the better performing algorithm for various known and unseen DVRP instances. Based on simulation results, we were able to derive characteristic DVRP instances from static problem instances, and developed and calculated features for these instances. The features in combination with the simulation results were the basis for our developed classification and regression models, which we trained to predict algorithm performance. The prediction of algorithm performance is the base for our successful autonomous algorithm selection. Currently, we only consider two different algorithms which perform similarly for different problem instances, compare Section 4. Future work will be the training and testing of classification or regression models based on simulation results of different DVRP algorithms and also of solving frameworks using different parameterization. Currently, a new DVRP solution approach based on the results from Mayer et al. (2017) is developed by us. For our performance prediction models, currently only 4 different structured problem instances are considered, compare Figure 2. The idea is to build a stronger basis for our performance prediction by considering more problem instances. We plan to include the very artificial structured VRP instances introduced in Golden et al. (1998) in our research. Another area of our future research considers the problem features. Currently, 16 features to describe the DVRP instances are used for our research. We have to investigate whether additional features lead to better performance prediction and thereby to better

algorithm selection. Beside considering more instances and more algorithms or features, the question occurs which evolved instances are good for training and testing purposes, compare Section 3. Currently, we are not verifying if an evolved instance is a good representative of the performance class. In future work, we have to find a metric which provides information about how good a certain instance represents a certain algorithm performance. A general future task is a deeper comparison between the classification and regression models. The predictions from the regression models are varying stronger, compare Table 2. But, the performance of the regression models is on average better for unseen problem instances. But, note that the classification models have limited explanatory power for instances where the different classification models predict the same performance classes. Even if the true algorithm results would vary strongly (compare Figure 3), if they are predicted to be in the same class, no reasonable algorithm selection can be done. A solution would be to have more classification models based on the results from more algorithms or to introduce more performance classes. But, more classes might lead to more wrong predictions for instances near to the new performance class borders, compare Section 3. In general, the artificially defined performance class borders which are needed for classification models lead to difficulties, e.g., on how to define them and on how to distribute the instances between them.

REFERENCES

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

Basheer, I. A., and M. Hajmeer. 2000. "Artificial Neural Networks: Fundamentals, Computing, Design, and Application". *Journal of Microbiological Methods* 43(1):3–31.

Bent, R. W., and P. Van Hentenryck. 2004. "Scenario-based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers". *Operations Research* 52(6):977–987.

Bridle, J. S. 1990. "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition". In *Neurocomputing*, edited by F. F. Soulié and J. Hérault, 227–236. Berlin, Heidelberg: Springer.

Burke, E., G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. 2003. "Hyper-Heuristics: An Emerging Direction in Modern Search Technology", In *Handbook of Metaheuristics*, edited by F. Glover and G. A. Kochenberger, 457–474. Boston, MA: Springer US.

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

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

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

Clarke, G., and J. W. Wright. 1964. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". *Operations research* 12(4):568–581.

Dantzig, G. B., and J. H. Ramser. 1959. "The Truck Dispatching Problem". *Management Science* 6(1):80–91.

Garrido, P., and M. C. Riff. 2010. "DVRP: A Hard Dynamic Combinatorial Optimisation Problem Tackled by an Evolutionary Hyper-heuristic". *Journal of Heuristics* 16(6):795–834.

Golden, B. L., E. A. Wasil, J. P. Kelly, and I.-M. Chao. 1998. "The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results". In *Fleet Management and Logistics*, edited by T. G. Crainic and G. Laporte, 33–56. Boston, MA: Springer US.

Gruler, A., A. Klüter, M. Rabe, and A. A. Juan. 2017. "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 S. Wenzel and T. Peter, 129–138. Kassel: kassel university press.

Juan, A. A., J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira. 2015. "A Review of Simheuristics: Extending Metaheuristics to Deal with Stochastic Combinatorial Optimization Problems". *Operations Research Perspectives* 2:62–72.

Juan, A. A., J. Faulin, J. Jorba, D. Riera, D. Masip, and B. Barrios. 2011. "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* 62(6):1085–1097.

Kilby, P., P. Prosser, and P. Shaw. 1998. "Dynamic VRPs: A Study of Scenarios". *University of Strathclyde Technical Report*:1–11.

Kingma, D. P., and J. Ba. 2015. "Adam: A Method for Stochastic Optimization". In *Proceedings of the International Conference on Learning Representations (ICLR)*. May 7th-9th, San Diego, CA, USA.

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

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

Lehmann, E. L., and G. Casella. 2006. *Theory of Point Estimation*. 2rd ed. New York: Springer Science & Business Media.

Lin, S., and B. W. Kernighan. 1973. "An Effective Heuristic Algorithm for the Traveling-salesman Problem". *Operations Research* 21(2):498–516.

Lund, K., O. B. Madsen, and J. M. Rygaard. 1996. "Vehicle Routing with Varying Degree of Dynamism". Technical Report IMM-REP-1996-1, IMM Institute of Mathematical Modelling, Lyngby, Denmark.

Mayer, T., T. Uhlig, and O. Rose. 2016. "An Open-source Discrete Event Simulator for Rich Vehicle Routing Problems". In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, edited by R. Rossetti and D. Wolf, 1305–1310. New York, USA: IEEE.

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

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

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

Muggeo, V. M. 2003. "Estimating Regression Models with Unknown Break-points". *Statistics in medicine* 22(19):3055–3071.

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

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

Pillac, V., M. Gendreau, C. Guéret, and A. L. Medaglia. 2013. "A Review of Dynamic Vehicle Routing Problems". *European Journal of Operational Research* 225(1):1–11.

Psaraftis, H. N. 1980. "A Dynamic Programming Solution to the Single Vehicle Many-to-many Immediate Request Dial-a-ride Problem". *Transportation Science* 14(2):130–154.

Psaraftis, H. N., M. Wen, and C. A. Kontovas. 2016. "Dynamic Vehicle Routing Problems: Three Decades and Counting". *Networks* 67(1):3–31.

Rice, J. R. 1976. "The Algorithm Selection Problem". *Advances in Computers* 15:65–118.

Ridge, E., and D. Kudenko. 2007. "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 C. Cotta and J. Van Hemert, 198–209. Berlin, Heidelberg: Springer.

Ritzinger, U., J. Puchinger, and R. F. Hartl. 2016. "A Survey on Dynamic and Stochastic Vehicle Routing Problems". *International Journal of Production Research* 54(1):215–231.

Schröder 2018. "Jsprit". https://github.com/graphhopper/jsprit. Accessed July 2nd, 2018.

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

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

Smith-Miles, K. A. 2009. "Cross-disciplinary Perspectives on Meta-learning for Algorithm Selection". *ACM Computing Surveys (CSUR)* 41(1):6.

Swingler, K. 1996. *Applying Neural Networks: A Practical Guide*. London: Academic Press.

Twomey, J. M., and A. E. Smith. 1997. "Validation and Verification". In *Artificial Neural Networks for Civil Engineers: Fundamentals and Applications*, edited by N. Kartam et al., 44–64. New York, NY, USA: American Society of Civil Engineers.

Uhlig, T. 2015. *Self-Replicating Individuals*. München: Verlag Dr. Hut.

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

Wilson, N. H., and N. J. Colvin. 1977. "Computer Control of the Rochester Dial-a-ride System". Technical Report 77-22, Massachusetts Institute of Technology, Center for Transportation Studies, Cambridge, Massachusetts.

Wolpert, D. H., and W. G. Macready. 1997. "No Free Lunch Theorems for Optimization". *IEEE Transactions on Evolutionary Computation* 1(1):67–82.

AUTHOR BIOGRAPHIES

THOMAS MAYER is working as Research Assistant and PhD student at Universität der Bundeswehr as a member of the scientific staff of Prof. Dr. Oliver Rose. His focus is on solving vehicle routing problems, and algorithm selection based on machine learning techniques. He has received his M.S. degree in Computer Science from Dresden University of Technology. His email address is thomas.mayer@unibw.de.

TOBIAS UHLIG is a postdoctoral researcher at the Universität der Bundeswehr München, Germany. He holds a M.Sc. degree in Computer Science from Dresden University of Technology and a Ph.D. degree in Computer Science from the Universität der Bundeswehr München. His research interests include operational modeling and heuristic optimization. He is a member of the ASIM and the IEEE RAS Technical Committee on Semiconductor Manufacturing Automation. His email address is tobias.uhlig@unibw.de.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Department of Computer Science of the Universität der Bundeswehr Munich, Germany. He received a M.S. degree in Applied Mathematics (1992) and a Ph.D. degree in Computer Science (1997) from Würzburg University, Germany. His research focuses on the operational modeling, analysis, and material flow control of complex manufacturing facilities, in particular, semiconductor factories and assembly systems. He is a member of INFORMS Simulation Society, ASIM (German Simulation Society), and GI (German Computer Science Society). Currently, he is member of the board of the ASIM and the ASIM representative at the Board of Directors of the WSC. His email address is oliver.rose@unibw.de.