Feature Extractors for Describing Vehicle Routing Problem Instances

Jussi Rasku[1], Tommi Kärkkäinen[1], and Nysret Musliu[3]

1 **Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35, FI-40014, Finland**
`jussi.rasku@jyu.fi`
2 **Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35, FI-40014, Finland**
3 **Institute of Information Systems, Vienna University of Technology, A-1040 Vienna, Austria**

Abstract

The vehicle routing problem comes in varied forms. In addition to usual variants with diverse constraints and specialized objectives, the problem instances themselves – even from a single shared source – can be distinctly different. Heuristic, metaheuristic, and hybrid algorithms that are typically used to solve these problems are sensitive to this variation and can exhibit erratic performance when applied on new, previously unseen instances. To mitigate this, and to improve their applicability, algorithm developers often choose to expose parameters that allow customization of the algorithm behavior. Unfortunately, finding a good set of values for these parameters can be a tedious task that requires extensive experimentation and experience. By deriving descriptors for the problem classes and instances, one would be able to apply learning and adaptive methods that, when taught, can effectively exploit the idiosyncrasies of a problem instance. Furthermore, these methods can generalize from previously learnt knowledge by inferring suitable values for these parameters. As a necessary intermediate step towards this goal, we propose a set of feature extractors for vehicle routing problems. The descriptors include dimensionality of the problem; statistical descriptors of distances, demands, etc.; clusterability of the vertex locations; and measures derived using fitness landscape analysis. We show the relevancy of these features by performing clustering on classical problem instances and instance-specific algorithm configuration of vehicle routing metaheuristics.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.1.6 Optimization, G.1.10 Applications, I.2.6 Learning, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Metaheuristics, Vehicle Routing Problem, Feature extraction, Unsupervised learning, Automatic Algorithm Configuration

Digital Object Identifier 10.4230/OASIcs.SCOR.2016.7

1 Introduction

The quality and the required computational effort of algorithmically optimized vehicle routing solutions are heavily dependent on the problem instance, the solution method, and using the right parameters for the algorithms [5]. Fortunately, it has been shown that automatic algorithm configuration and algorithm selection can be used to improve the solver performance. Thus, in order to make routing algorithms more robust and adaptive, we propose applying machine learning to help the algorithms more effectively adapt to the problem being solved. However, as a prerequisite, we need a way to describe the problem instances to the learning algorithms.

The vehicle routing problem (VRP) can be considered to be a generalization of the traveling salesman problem (TSP). Therefore, studies where TSP instances are described, e.g. [20, 8, 12, 6, 13], are highly relevant. Smith-Miles and van Hemert [20] proposed 12 features for predicting the most suitable optimization algorithm for a TSP instance. The feature set is well-rounded containing features derived from the distance matrix, clustering, nearest neighbors, and geometry of an instance. Kanda et al. [8] had a similar goal, but they relied only on features based on the problem size and statistical description of the distance matrix, whereas Mersmann et al. [12] proposed a set of 47 features in order to build a model that could be used to discriminate between hard and easy TSP instances. Hutter et al. [6] proposed a set of new approaches such as describing minimum spanning trees, ruggedness, and probing with TSP solvers. Probing involved analyzing and describing the solution attempts with a heuristic and branch-and-cut solvers. Pihera and Musliu [13] further extended this feature set, which allowed algorithm selection for a TSP instance.

Literature of VRP descriptors is scarce. The only studies on VRP feature extraction from the machine learning perspective we are aware of are the dissertation of Steinhaus [22] and algorithm performance prediction in [25]. Steinhaus [22] explores the use of a self organizing maps in solving VRPs and in algorithm selection. She proposes 23 features specifically for VRP problems and explores the discrimination power of this feature set across 102 VRP benchmark problems. Most features she proposed are based on earlier literature on describing TSPs, but they are complemented with features describing the demand distribution of the nodes, vehicle capacity, and their relations. Studies from a VRP *fitness landscape analysis* perspective, e.g. [21, 14, 25], do exist, but as these metrics are mainly used to gain deeper understanding of the problem, they need to be adapted before they can be used for performance prediction or algorithm selection. This was the approach chosen by Ventresca et al. [25].

In this article, a set of feature extractors gathered from the aforementioned sources is adapted to describe capacitated vehicle routing problem (CVRP) instances. Our goal is to recognize problem types and better understand instance properties that may affect solving them. A set of features that is this comprehensive has not been previously used to describe vehicle routing problems. Moreover, the feature set is validated experimentally with clustering of benchmark instances, automatic algorithm configuration [5], and instance specific algorithm tuning [7].

Our contributions are threefold: First, we give a review on feature extraction of vehicle routing problems. Second, proposed features are used in automatic configuration of three metaheuristic CVRP solvers to prove their usefulness for self-adaptive and learning solution techniques. Finally, we do clustering on 168 well known CVRP benchmark instances and make observations on their similarities. To the best of our knowledge, this is the first study that proposes the use of features acquired by probing CVRP instances with exact and heuristic solution methods. This is also the first study to explore the possibility of using features to improve the performance of automatic configuration of vehicle routing algorithms.

This paper is organized as follows: In Section 2 the automatic algorithm configuration problem is defined. Section 3 introduces the vehicle routing problem in detail, with handling of common solution approaches. It is followed by listings of feature extractors and descriptors for these problems, also including those presented in this study. Section 4 describes the experimental setup and the results for verifying the proposed feature set. Finally, we conclude our study in Section 5.

2 Instance Specific Algorithm Configuration Problem

The task of automatic algorithm configuration (AAC) involves the off-line task of finding a "good" set of parameter values, or *a parameter configuration*, for a *target algorithm* in a way that the algorithm achieves the best possible performance. It is critical to use a representative set of problem instances when configuring the algorithm parameters. This ensures that the performance advantage manifests also on new, previously unseen, instances.

If a good generalized performance is needed and the problem set is not homogeneous, i.e. the instances are very different from each other, the use of AAC may even be disadvantageous: a parameter configuration may enable an algorithm to perform well on some instances, but be inferior to algorithm defaults on another. One possible solution in a situation like this is to use instance specific algorithm configuration as described e.g. in Kadioglu et al. [7]. The idea is to configure the parameters for each group of mutually similar problem instances separately, and when a new problem instance needs to be solved, the automatically configured parameters of the most similar instance group is used. For a study on instance specific algorithm configuration of a TSP metaheuristic we refer to [18].

The task of algorithm selection is closely related to AAC. In algorithm selection, the problem instance properties are used to choose the algorithm with best predicted performance. Usually the algorithm is selected out of a portfolio, and the model for algorithm performance is built earlier during an off-line learning phase. The approach has proven successful: during the last decade many state-of-the-art results in combinatorial optimization competitions have been achieved using algorithm selection from an algorithm portfolio [27, 10].

Please note that both algorithm selection and instance specific algorithm configuration need a way to describe the problem instances. Therefore, a good set of feature extractors is a critical prerequisite for employing these learning meta-optimization techniques. It is necessary to experimentally discover which features can characterize a problem set in such a way they capture properties relevant to a) solving the problems b) configuring algorithm parameters c) recognizing a set of mutually similar problems that can share a configured parameter configuration, and d) ability to predict algorithm performance.

3 The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) involves finding optimal routes for *vehicles* leaving from a *depot* to serve number of *clients*. Each client must be visited exactly once by exactly one vehicle. Each vehicle must leave from the depot and return there after serving the clients on its *tour*. There are numerous variants of VRPs, each with their own additional constraints [23]. In this study, only the classic Capacitated Vehicle Routing Problem (CVRP) is considered. In the CVRP, each of the identical vehicles has a maximum carrying capacity of Q that cannot be exceeded at any point of the tour. Each of the clients, indexed with i, have a demand q_i that has to be within $0 < q_i \leq Q$. The number of vehicles, denoted by k, is the primal minimization target, followed by the total travel distance of the k vehicles. Extending this notation, the CVRP can be written in a graph formulation adapted from Toth and Vigo [23] as follows: Let $V = \{0, \ldots, n\}$ be the set of vertices where the depot has the index 0 and where the rest correspond to the clients. The size of the problem is denoted by $N = |V|$. Let $E = \{(0,1), \ldots, (i,j), \ldots, (n-1,n)\}$ be the set of edges, where $i, j \in V, i \neq j$. Therefore, the graph $G = (V, E)$ is complete with each edge $e = (i,j) \in E$ having an associated non-negative weight c_{ij} that is the cost of traversal from vertex i to j. The weights can be also given as a distance matrix D. For each of the edges $(i,j) \in E$ there is a binary decision variable x_{ij} to decide whether the edge is traversed.

3.1 On Solving Vehicle Routing Problems

The solution approaches of VRPs can be divided into two main families: exact and heuristic methods. Heuristic methods are often augmented with metaheuristics to avoid entrapment in the first local optima the search encounters. Laporte [9] further divides heuristic methods into constructive and improvement heuristics. Constructive heuristics insert unassigned clients on the routes, and improvement heuristics improve the solution quality through small moves until no improving steps can be taken. Improvement heuristics can be seen as a building block of the Local Search (LS), a key element in modern metaheuristics.

The (meta)heuristic approach is the most feasible approach when solving larger CVRP problems. However, exact methods are still relevant as the (meta)heuristic methods make no guarantees in reaching the globally optimal value. According to Lysgaard et al. [11], the most promising exact solution technique for CVRP appears to be branch-and-cut (BnC). In BnC cutting planes are iteratively added to a relaxed linear programming model to ultimately narrow down on the global optimum.

3.2 Descriptors for the Problem

The features for routing problem instances are usually calculated either using the distance matrix or the 2D coordinates. Therefore, to calculate all features, both the node coordinates and the distance matrix need to be known. If D was not given in an instance file, a distance matrix was produced using the depot and client coordinates. Likewise, if a benchmark instance provided only a distance matrix, we used multidimensional scaling (MDS) [2] to generate x and y coordinates for the instance. We also followed the example of Smith-Miles and van Hemert [20] and scaled the coordinates into the $[(0,0),(400,400)]$ rectangle to make the geometrical features comparable between problem instances. However, we retain the shape (scale) of the problem when normalizing the problem to avoid distortion of the distance matrix, i.e. we maintained the x/y ratio. To maintain the connection between the coordinates and D, we scaled the distance matrix D using the same multiplier as with coordinates. This preprocessing produces a commensurable distance matrix D^n and coordinate set P^n that can be used to calculate geometrical and graph features.

Table 1 (p. 5) presents the CVRP feature extractors used in this study. The features proposed in this study are marked with bold typeface. The table also shows how many feature values each extractor produces. Usually the features are statistical descriptors explaining the distribution of measured values. If the number of statistical descriptors is five, it includes statistical moments (mean, standard deviation, skewness and kurtosis) and coefficient of variation; whereas if 11 descriptors are given, the former are complemented by minimum, maximum, median, number of modes, frequency of the mode value, and the mode itself (or average of modes). An even more complete set of 14 descriptors adds quartiles.

Table 1a. The first feature set is for describing the node distribution on a 2D plane. The most often used feature involves statistically describing the distance matrix (cost matrix, without the diagonal). Smith-Miles and van Hemert [20] used the standard deviation, which Kanda et al. [8] and Hutter et al. [6] complemented with a more comprehensive set of statistical descriptors. We normalized the distance matrix to the rectangle $[(0,0),(400,400)]$, similarly to [20], and calculate 11 statistical descriptors for the distance distribution. Smith-Miles and van Hemert [20] also proposed counting the distinct distances found in the distance matrix using different precision. We used four levels of precision, like in [6]. Also, the centroid of the coordinates and the euclidean distance from each point to the centroid were calculated. The average of these distances is the "radius" feature from [20].

Table 1 The feature extractors for CVRPs, grouped by type.

(a) Node distribution features

ID	Feature	#
ND1	Distribution of distance matrix calues [20, 8, 6]	11
ND2	Fraction of distinct distances (with 1,2,3,4 decimals) [20, 6]	4
ND3	Centroid of the nodes (x,y) [20]	2
ND4	Distance to the centroid [20]	5
ND5	# of clusters (abs.,rel.) [20]	2
ND6	# of core, edge and outlier cluster points (rel.) [20]	3
ND7	Reach of the clusters [20]	5
ND8	Normalized cluster sizes [6]	5
ND9	**Silhouette coefficient**	1
ND10	Minimum bottleneck cost [6]	5

(b) Minimum spanning tree (MST) features

ID	Feature	#
MST1	MST edge cost [12, 6]	5
MST2	MST node degree [12, 6]	5
MST3	MST depth **from the depot**	5

(c) Local search (LS) probing features

ID	Feature	#
LSP1	Solution quality after construction phase [6]	5
LSP2	Solution quality after LS [6]	5
LSP3	Improvement per LS step [6]	5
LSP4	LS steps to local minimum [6]	5
LSP5	Distance of local minima [6]	5
LSP6	% for edges in local optima [6]	5
LSP7	Solution edge lengths per quartile (**5** $\times$ **4** quartiles) [13]	**20**
LSP8	Segment length [13]	5
LSP9	Segment edge count [13]	5
LSP10	Segment edge length [13]	5
LSP11	Intra-tour intersections [13]	5
LSP12	**Autocorrelation length**	5

(d) Branch-and-cut probing features

ID	Feature	#
BCP1	Improvement per added cut [6]	5
BCP2	Ratio between upper and lower bound [6]	1
BCP3	Solution value after probing [6]	1
BCP4	Lower bound [6]	1

(e) Geometric features

ID	Feature	#
G1	Area of the enclosing rectangle ("squareness") [20, 6]	1
G2	Convex hull (CH) area [12]	1
G3	Ratio of points on the hull [12]	1
G4	Distance of enclosed points to the CH contour [13]	11
G5	Edge lengths of the CH [13]	11

(f) Nearest neighborhood (NN) features

ID	Feature	#
NN1	Distance to 1st NN [20, 6]	5
NN2,9,15	Node input degree in directed kNN graph (DkNNG) for $k \in \{3, 5, 7\}$ [13]	14
NN3,10,16	# of strongly connected components (SCCs) in DkNNG	11
NN5,11,17	Size of SCCs in DkNNG [13]	11
NN6,12,18	# of Weakly Connected Components (WCCs) in DkNNG	11
NN7,13,19	Size of WCCs in DkNNG [13]	11
NN8,14,20	Ratio of SCCs/WCCs [13]	1
NN21	Angle between edges to two NNs [12, 13]	11
NN22	Cosine similarity between edges to two NNs [13]	11

(g) VRP specific features

ID	Feature	#
DC1	Number of clients [8]	1
DC2	The depot location (x, y) [22]	2
DC3	**Distance between the centroid and the depot**	1
DC4	**Client dist. to the depot**	5
DC5	Client Demands [22]	5
DC6	Ratio of total demand to total capacity (the "tightness") [22]	1
DC7	Ratio of max. cluster demand to vehicle capacity [22]	1
DC8	Ratio of cluster outlier to overall demand [22]	1
DC9	Ratio between the largest demand and the capacity [22]	1
DC10	Average number of clients per vehicle [22]	1
DC11	Minimum number of trucks [22]	1

Some heuristics rely heavily on the existence of clusters. Therefore, features capturing this aspect are expected to be useful in algorithm selection. DBSCAN clustering has been used, at least, in [20, 8, 12, 6, 13, 22] to extract features for routing problem instances. We calculated features for cluster count (absolute and relative to N); cluster size; and a relative number of core, edge and outlier points. Mersmann et al. [12] used three different values for the ϵ (maximum allowed distance for two points belonging to a same cluster), while Steinhaus [22] experimented with four alternative methods to find a good ϵ value. In our study we decided to use the minimum cluster size of 4, and with that $\epsilon = A_{\text{bb}}/(\sqrt{N}-1)$, which is an approximation of the 4th nearest neighbor distance if the nodes are assumed to be uniformly distributed on a lattice within a square with an area of A_{bb} [22]. To include a feature measuring the quality of the DBSCAN clustering, we propose the silhouette score [19] as a novel addition to the feature set.

The node distribution features are completed with the Minimum Bottleneck Cost (MBC) as proposed by [6]. It is used to describe the clusterability of TSP instances. The bottleneck cost is defined to be the weight of the longest edge on a path from node i to j, $i \neq j$. We get the minimum bottleneck cost by taking a minimum of bottleneck costs over all possible paths from node i to j. By calculating the bottleneck cost for all possible node pairs $i, j \in V$, $i \neq j$, we get a distribution that can then be described with statistical moments.

Table 1b. A minimum spanning tree (MST) was calculated for the fully connected normalized graph G^n. As suggested in [12, 6], the distribution of edge costs and node degrees of the MST were described using statistical moments. Mersmann et al. [12] included the spanning tree node depth as well, which we adapted for the VRP by calculating it with the depot as the root. We omitted the sum of the MST tree cost proposed in [12], as it can be inferred from the average MST cost.

Table 1c Probing features are computed with a solution attempt on a problem instance. An algorithm is run for predefined time or steps and the trajectory of the search is recorded. The approach is general and applicable to a variety of problems. Probing has been shown to be useful, e.g. for predicting the performance of an algorithm [6].

To adapt the TSP LS probing features from [13], we used the VRPH heuristic search algorithm library [3], or more specifically, its `vrp_init` application that is based on the Clarke-Wright construction heuristic. It was modified to accept a shape parameter γ that affects the savings calculation [28]. The parameter can be selected randomly to produce varied initial solutions. After construction, the solution is improved with intra-route multi-neighborhood search using best accept strategy with one-point-move, two-point-move and two-opt local search heuristics [3] until no improving move is found.

By repeating the probing 20 times, we could calculate the statistical descriptors in Table 1c. Some of the features closely resemble those we have used previously to validate visualization technique for VRPs with solution space analysis (SSA) [16]: the first is the distribution of Manhattan distances between the local optimum solutions (`LSP5`), calculated from the differences in edge traversal decision variable values between the solutions. This feature is a measure of the multimodality and indicator for the existence of a "big-valley" structure [14]. The second SSA feature `LSP6` describes the distribution of probabilities of all edges in locally optimal solutions, which aims to reveal the existence of a backbone [26], that is, a common structure between good solutions.

The features `LSP8-10` involve the concept of a segment. A segment is a continuous path of consecutive edges on a tour, from which the longest edges are removed as specified in [13]. Pihera and Musliu [13] also proposed another extension to the set of local search

features, which is the number of intra-tour intersections, i.e. the times the edges of a tour cross each other.

The final local search probing feature is the autocorrelation length λ_{ACL} of a random walk through a series of best-accept one-point-move neighborhoods (heuristic described e.g. in [3]). This closely relates to the autocorrelation coefficient used in [6]. For calculating the autocorrelation length we used a method adapted from [21] and [4], with a random walk length of $2N$. The walk is repeated and the length calculated 10 times.

Table 1d. Besides heuristics, also exact solvers can be used to probe the problem. We used the open source mixed-integer programming package SYMPHONY 5.6.15 and its VRP application that can solve CVRPs [15]. Unfortunately, we were not able to compile the VRP application with heuristics support for this version of SYMPHONY. Therefore, the upper bound is set only after the first feasible solution is found. Because of this, is possible that the feature `BCP2` is left undefined for the larger instances if no feasible solution is found. SYMPHONY also requires the number of vehicles k as an input when solving an instance. If k was not known we divided the total demand with some margin (+5%) with the vehicle capacity Q to get the value for k, i.e. $k = \lceil 1.05 \sum q_i/Q \rceil$. For branch-and-cut probing we used a wall time cutoff of 3.0 seconds.

Table 1e. Geometric features try to capture information of the overall shape of the problem. The area of an enclosing rectangle, when normalized with the area of the scaled problem, describes the "squareness" of the problem. Mersmann et al. [12] suggested two features concerning the convex hull: the hull area and the fraction of nodes that are on the hull contour. According to their experiments, convex hull features allow accurate separation of easy and hard TSP instances. Pihera and Musliu [13] added statistical descriptors for distances of inner nodes to the hull contour. It is assumed that the more evenly distributed the nodes are inside the convex hull, the more difficult it is to solve. Therefore, all these were included in our feature set.

Table 1f. Because heuristic solution methods operate by navigating through the search space using a local search neighborhood, the Nearest Neighbors (NN) of the nodes can offer important insight to the structure of the problem. In our study, the distribution of the 1st nearest neighbor distances over all nodes is statistically described as done in [20, 6, 13]. We also included the extended nearest neighbor features presented in [13], which involve building a directed graph by taking only $k \in 3, 5, 7$ shortest edges for each node from the complete normalized graph G^n. The node degree, number and size of strongly and weakly connected components, and their ratios are calculated and statistically described.

Table 1g. In describing the demands and capacity, we followed [22]. As an extension to the VRP specific features, we propose measuring the distance between the depot and the centroid of the client points. Also, describing the shape of the distribution of distances from clients to the depot is included in our feature set. Furthermore, the size of the problem (number of clients) is included here. Refer to [22] for details on these features.

In addition to the features presented in Table 1, we recorded the per instance feature computation time as proposed in [6]. These are reported as timing features `T1-T9` that match the feature groups (Tables 1a–1g), with the exception of the autocorrelation and bottleneck cost features, which are timed separately.

To summarize this section, we have adapted and proposed 76 feature extractors for CVRPs which generate 386 features in total. The feature extractors were implemented in Python version 2.7.10, with the aid from numerical libary Numpy (version 1.9.2), machine learning library Scikit-learn (version 0.16.1), and statistical library Scipy (0.15.1). VRPH and SYMPHONY were built with GNU g++ 5.3.0 compiler from the Mingw-w64 project.

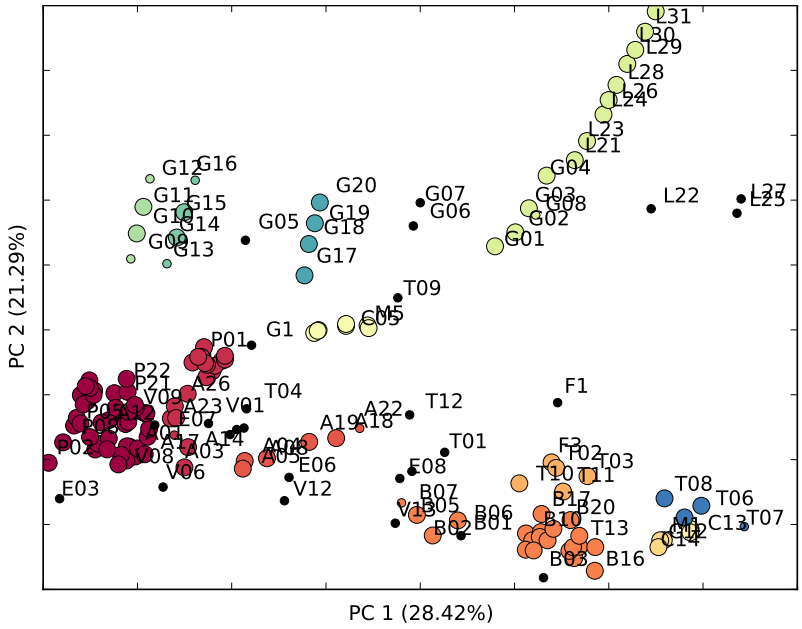

Figure 1 The clustering of the benchmark instances. Black dots are non-clustered outliers. The plot axes are the first two principal components, with the ratio of explained variance given in parenthesis.

All feature extraction in this study was done on a laptop with dual-core 2.53 GHz Intel Core i5 520M processor, 8 GB of memory and 64-bit Windows 7 Enterprise operating system.

4 Experimental Evaluation of the Features

4.1 Clustering

To evaluate the quality of the proposed feature set, we computed the 386 features for each of the 168 problem instances in CVRPLIB, which is a collection of CVRP benchmark instances [24]. However, the high dimensionality of the resulting data had to be addressed before clustering. Hutter et al. [5] suggests using principal component analysis (PCA) to reduce the computational complexity when building a surrogate model for the automatic algorithm configuration tool SMAC. We share some of the concerns regarding the computational cost. However, in our case a larger issue is the curse of dimensionality, where the space volume grows very rapidly as the dimensionality increases. This makes the dataset too sparse to provide a representative sample of the high dimensional space. A related problem is the irrelevancy of the distance metric in high dimensional data, where all data points seem to be similarly close to each other [1]. To overcome these issues, we reduced the dimensionality of the feature space with PCA.

To do the actual clustering, the feature data was first normalized by scaling all features independently to a range $[0.0, 1.0]$. Then, PCA was applied to bring the dimensionality of the data down from 386 to 7 following the example of [5]. These seven principal components together explain 71 % of the overall variance in the data. Finally, to do the unsupervised learning, we used the DBSCAN with a minimum cluster size of 3. The ϵ parameter was set to 0.20 through experimentation. Resulting clusters for the 168 benchmark instances are presented in Figure 1.

The clusters in the lower left corner seem to be the small-to-medium easy-to-solve instances. Unsurprisingly, the A and P sets overlap, as P is based on A. The difference between A and B is that clients in A are uniformly generated whereas in B they are clustered. Also the Taillard (T) and Fisher (F) sets contain clustered clients, which can be observed as an overlap with the set B. Interestingly, the benchmark set Golden (G) is separated into four clusters and some outliers. The benchmark set contains points in geometric shapes like stars, squares, circles and rays, and it seems that our features are able to discriminate between these. The Li (L) and Golden benchmark sets are similar and clustering them together is expected. For a more accurate analysis of clusters we would need to do a more extensive experimentation with the solvers, since probing does not necessarily allow reliable estimation of the hardness and computational difficulty of an instance. The complete list of problem instance abbreviations and the clustering in a table format, together with an interactive zooming visualization of the clustering, can be found from the supplementary online appendix at `http://users.jyu.fi/~juherask/features/`

4.2 Instance Specific Algorithm Configuration

As the automatic algorithm configuration targets, we used the three metaheuristic solvers provided by the VRPH package from Groër et al. [3]. Each solver employs different metaheuristic: Record-to-Record travel (VRPH-RTR, 6+8 free parameters), simulated annealing (VRPH-SA, 6+5), and ejection (VRPH-EJ, 6+3). We omit the descriptions of the algorithms and solver parameters and refer the reader to [3], whereas a detailed description of the automatic algorithm configuration setup can be found in [17].

As a configurator, we used SMAC [5]. SMAC is a state-of-the-art AAC method that alternates between fitting a random forest model to the observed behavior of the target algorithm, and using that model to predict the performance of generated parameter configuration candidates – evaluating only those that are most promising on the solver. SMAC offers an option to complement the problem instances with feature values, which are used when building and updating the random forest model. In our experiments this approach is called fSMAC. fSMAC already does PCA to the feature vectors, but as an additional preprocessing step we took 50 features that showed the highest correlation with the solution quality in heuristic and branch-and-bound probing. SMAC is not an instance specific algorithm configuration tool like ISAC from Kadioglu et al. [7], but we can follow a similar scheme to create IS-fSMAC. This variant uses k-Means clustering on the preprocessed feature data to split the problem instance set to subsets. These subsets supposedly share similar solving characteristics and can be configured separately.

In our configuration experiments we used a set of 14 instances taken with stratified sampling from the CVRPLIB set A. The problem set is the same one that we used in [17], which makes it possible for the interested reader to compare the proposed approach against other configurators. Also, each configuration task was run with three different evaluation budgets (EBs): 100, 500, and 1000 time capped (10 s) runs of the target algorithm. In the case of configuring the clustered instances, the budget was distributed according to the cluster size. Because the algorithms are stochastic, the experiments were repeated 10 times.

The results of the configuration tasks are presented in Table 2. The use of features seems beneficial, especially with a budget of 100. This is unsurprising, as the use of features is expected to provide more initial information when building the surrogate model of the parameter-solution quality response surface. Especially VRPH-SA target seems to benefit from using the features. The advantage gained by using features is smaller for VRPH-EJ and VRPH-RTR targets, but the effect still exists. However, the results of instance specific

Table 2 Median tuning results for VRPH metaheuristics. Results are given as percentages from the aggregated best known solution (relative optimality gap). The best known solution values are from CVRPLIB. Statistically better results are in bold ($p < 0.05$ with Bonferroni adjustment). If no single best was found, a test for a best pair was made.

Target	**VRPH-SA**			**VRPH-EJ**			**VRPH-RTR**		
Defaults	0.83 (0.12)			0.50 (0.13)			1.42 (0.07)		
Method \ EB	**100**	**500**	**1000**	**100**	**500**	**1000**	**100**	**500**	**1000**
SMAC	**0.40**	0.29	0.26	0.39	**0.35**	0.34	**0.16**	0.09	0.10
	(0.11)	(0.08)	(0.06)	(0.07)	**(0.04)**	(0.04)	**(0.05)**	(0.01)	(0.02)
fSMAC	**0.39**	0.26	**0.23**	**0.36**	0.36	0.35	**0.15**	0.09	**0.07**
	(0.15)	(0.06)	**(0.05)**	**(0.06)**	(0.05)	(0.06)	**(0.06)**	(0.04)	**(0.03)**
IS-fSMAC	0.56	0.27	0.25	**0.35**	**0.33**	0.34	0.19	0.09	0.09
	(0.11)	(0.07)	(0.05)	**(0.08)**	**(0.07)**	(0.06)	(0.06)	(0.03)	(0.01)

parameter tuning are not as good as expected. The clustering to problem classes seems to be beneficial only for VRPH-EJ targets. It may be that the features are unable to capture the differences (unlikely), the clustering is handicapped by the curse of dimensionality (likely), or the evaluation budget split among clusters is too small for SMAC to converge to good parameter configurations (likely). Still, the most probable cause is the homogeneity of the problem set. All of the instances in the set A come from the same generator, thus showing similar solving characteristics. Additional experiments are needed to identify the largest factor preventing the instance specific tuning from giving comparable advantage to what has been reported in e.g. in [7]. Nonetheless, every resulting parameter configuration is superior compared to the defaults.

All automatic configuration was done on a computing server with 64 Intel(R) Xeon(R) CPU E7 2.67 GHz cores, and 1 TB of RAM running 64-bit OpenSUSE version 13.2 (codename Harlequin). We enforced a 10 second cutoff for all evaluations of the CVRP solver.

5 Conclusions

In this article, we set out to find feature extractors for capacitated vehicle routing problem (CVRP) instances, mostly by adapting Traveling Salesman Problem (TSP) descriptors from the literature. We implemented 76 feature extractors for almost every descriptor that had been reportedly used in algorithm selection and automatic algorithm configuration of routing algorithms and proposed some novel ones. The presented set of 386 features for CVRP is unparalleled in its extent. Additionally, we are not aware that probing with heuristic and branch-and-cut solvers has been previously used to produce features for CVRP meta-optimization. The suitability of these features was verified with feature assisted automatic algorithm configuration with the state-of-the-art tool SMAC. We also presented clustering of 168 well-known benchmark instances from the CVRPLIB collection. Clustering shows good discrimination ability between the known properties of these problems. However, a more complete analysis of the clustering is warranted to get novel insights.

We can conclude that automatic algorithm configuration can benefit from using the proposed features. Out of the tested CVRP metaheuristics, the simulated annealing (VRPH-SA) benefited the most. We also experimented with instance specific configuration, where it was possible to further improve the configured solver performance of the ejection metaheuristic (VRPH-EJ). However, the overall increase in performance when using an instance specific

algorithm configuration scheme was modest. This is probably due to our problem set being relatively small and homogeneous. Therefore, a more extensive experimentation with different targets, instance specific configurators, and problem sets is required to make a judgment on applicability of instance specific parameter configuration of vehicle routing solvers. Also, please note that in our automatic algorithm configuration experiments we did not test for over-tuning (cf. overfitting), which may manifest as poor generalizability of the configured parameter configuration.

Other future research topics include: Feature selection that should help us recognize the most useful features, as currently the high dimensionality of the feature vector seems to confuse unsupervised learning and algorithm configuration efforts. We would also like to extend our feature extractors to describe other well-known VRP variants such as vehicle routing problem with time windows (VRPTW) and pickup and delivery problems (PDP). This could potentially reveal new interesting similarities between the problem types and sets. We would also like to extend our study towards algorithm selection.

It has been shown that applying feature based machine learning approaches, such as the one presented here, in solving combinatorial optimization problems, can lead to significant improvements in on-line algorithm performance and resulting solution quality. Adapting this approach in solving VRPs has shown promise and warrants further research.

References

1 Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Bussche and Victor Vianu, editors, *Proceedings of Database Theory (ICDT 2001): 8th International Conference*, pages 420–434, Berlin, Heidelberg, 2001. Springer.

2 I Borg and P Groenen. Modern multidimensional scaling: theory and applications. *Journal of Educational Measurement*, 40(3):277–280, 2003.

3 Chris Groër, Bruce Golden, and Edward Wasil. A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2(2):79–101, April 2010.

4 Wim Hordijk. A measure of landscapes. *Evolutionary computation*, 4(4):335–360, 1996.

5 Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

6 Frank Hutter, Lin Xu, Holger H Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

7 Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC – instance-specific algorithm configuration. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *Proceedings of 19th European Conference on Artificial Intelligence (ECAI 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 751–756. IOS Press, 2010.

8 Jorge Kanda, Andre Carvalho, Eduardo Hruschka, and Carlos Soares. Selection of algorithms to solve traveling salesman problems using meta-learning. *International Journal of Hybrid Intelligent Systems*, 8(3):117–128, 2011.

9 Gilbert Laporte. What you should know about the vehicle routing problem. *Naval Research Logistics*, 54(8):811–819, 2007.

10 Marius Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. AutoFolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, 53:745–778, 2015.

11 Jens Lysgaard, Adam N Letchford, and Richard W Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

12 Olaf Mersmann, Bernd Bischl, Heike Trautmann, Markus Wagner, Jakob Bossek, and Frank Neumann. A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, 69(2):151–182, 2013.

13 Josef Pihera and Nysret Musliu. Application of machine learning to algorithm selection for TSP. In *Tools with Artificial Intelligence (ICTAI), IEEE 26th International Conference on*, pages 47–54. IEEE, 2014.

14 E. Pitzer, S. Vonolfen, A. Beham, M. Affenzeller, V. Bolshakov, and G. Merkuryeva. Structural analysis of vehicle routing problems using general fitness landscape analysis and problem specific measures. In *14th International Asia Pacific Conference on Computer Aided System Theory*, pages 36—38, 2012.

15 Ted K Ralphs. Parallel branch and cut for capacitated vehicle routing. *Parallel Computing*, 29(5):607–629, 2003.

16 Jussi Rasku, Tommi Kärkkäinen, and Pekka Hotokka. Solution space visualization as a tool for vehicle routing algorithm development. In Mikael Collan, Jari Hämälainen, and Pasi Luukka, editors, *Proceedings of the Finnish Operations Research Society 40th Anniversary Workshop (FORS40)*, volume 13, pages 9–12. LUT Scientific and Expertise Publications, 2013.

17 Jussi Rasku, Nysret Musliu, and Tommi Kärkkäinen. Automating the parameter selection in VRP: an off-line parameter tuning tool comparison. In William Fitzgibbon, A. Yuri Kuznetsov, Pekka Neittaanmäki, and Olivier Pironneau, editors, *Modeling, Simulation and Optimization for Science and Technology*, pages 191–209. Springer, 2014.

18 Jana Ries, Patrick Beullens, and David Salt. Instance-specific multi-objective parameter tuning based on fuzzy logic. *European Journal of Operational Research*, 218(2):305–315, 2012.

19 Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

20 Kate Smith-Miles and Jano van Hemert. Discovering the suitability of optimisation algorithms by learning from evolved instances. *Annals of Mathematics and Artificial Intelligence*, 61(2):87–104, 2011.

21 Peter F Stadler. Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20(1):1–45, 1996.

22 Meghan Steinhaus. *The Application of the Self Organizing Map to the Vehicle Routing Problem.* PhD thesis, University of Rhode Island, 2015.

23 Paolo Toth and Daniele Vigo, editors. *The vehicle routing problem.* SIAM, 2002.

24 E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, A. Subramanian, and T. Vidal. New benchmark instances for the capacitated vehicle routing problem. Technical report, UFF, Rio de Janeiro, Brazil, 2014.

25 Mario Ventresca, Beatrice Ombuki-Berman, and Andrew Runka. Predicting genetic algorithm performance on the vehicle routing problem using information theoretic landscape measures. In Martin Middendorf and Christian Blum, editors, *Proceedings of the Evolutionary Computation in Combinatorial Optimization: 13th European Conference (EvoCOP 2013)*, pages 214–225, Berlin, Heidelberg, 2013. Springer.

26 Toby Walsh and John Slaney. Backbones in optimization and approximation. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

27 Lin Xu and Kevin Leyton-brown. SATzilla : Portfolio-based algorithm selection for SAT. *Artificial Intelligence*, 32:565–606, 2008.

28 P.C. Yellow. A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, 21:281–293, 1970.