Instance Space Analysis of the Capacitated Vehicle Routing Problem with Mixture Discriminant Analysis

Danielle Notice
d.a.notice@lancaster.ac.uk
Lancaster University
Lancaster, United Kingdom

Hamed Soleimani
h.soleimani@latrobe.edu.au
LaTrobe University
Bundoora, Australia

Nicos G. Pavlidis
n.pavlidis@lancaster.ac.uk
Lancaster University
Lancaster, United Kingdom

Ahmed Kheiri
ahmed.kheiri@manchester.ac.uk
The University of Manchester
Manchester, United Kingdom

Mario Andrés Muñoz
munoz.m@unimelb.edu.au
The University of Melbourne
Parkville, Australia

Abstract

In this paper, we attempt a deeper understanding of the relative performance of two state-of-the-art metaheuristic solvers for the capacitated vehicle routing problem (CVRP). To this end, we employ a novel CVRP instance generator to expand the set of CVRP instances used to assess heuristics. This generator modifies existing problem instances using the outliers of node clusters to produce relevant new CVRP instances. We consider a large number of features to characterise each problem instance, and propose to use mixture discriminant analysis (MDA) to obtain both a low dimensional representation of the instance space and a classifier of algorithm performance. MDA has not been previously used in instance space analysis, and as a supervised dimension reduction method has the advantage that the tasks of dimension reduction and classification are handled in a unified framework (rather than two separate steps). The resulting predictive models perform as well as more complex classifiers that involve more tuning parameters and are computationally more expensive (like support vector machines). Our analysis highlights that the performance comparison between the two CVRP metaheuristics is nuanced and the best algorithm depends on the time budget, as well as certain key characteristics of the problem instance.

CCS Concepts

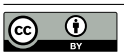
• **Computing methodologies** → *Supervised learning by classification*; • **General and reference** → *Empirical studies.*

Keywords

Capacitated vehicle routing problem, instance space analysis, algorithm selection, supervised dimension reduction, discriminant analysis

ACM Reference Format:

Danielle Notice, Hamed Soleimani, Nicos G. Pavlidis, Ahmed Kheiri, and Mario Andrés Muñoz. 2025. Instance Space Analysis of the Capacitated Vehicle Routing Problem with Mixture Discriminant Analysis. In *Genetic and Evolutionary Computation Conference (GECCO '25), July 14—18, 2025, Málaga, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3712256.3726405



This work is licensed under a Creative Commons Attribution 4.0 International License.
GECCO '25, July 14–18, 2025, Málaga, Spain
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1465-8/2025/07
https://doi.org/10.1145/3712256.3726405

1 Introduction

Due to its importance in real world applications, the vehicle routing problem (VRP) and its variants are among of the most extensively studied combinatorial optimisation problems. In this work we focus on the capacitated VRP (CVRP). The typical size of CVRP instances in real-world applications is such that exact algorithms are not feasible, and hence heuristics are widely used in practise. In the literature on CVRP heuristics (as in other areas of operations research [30]) when a new algorithm is proposed it is typically shown to outperform (on average) existing algorithms on a set of test instances. In this work we employ instance space analysis [33] to develop a deeper understanding of performance differences between two state-of-the-art CVRP metaheuristics: Hybrid Genetic Search with advanced diversity control (HGS) [37], and the Fast Iterated local search Localized Optimization (FILO) algorithm [1].

To this end, we first propose a novel CVRP instance generator and use it to expand the set of test instances commonly used. This generator modifies existing problem instances by first clustering the customers, and then adding new customers based on those clusters. The instances thus generated are adequately similar to those in the original set to be considered relevant but are clearly not identical. To understand under what conditions HGS outperforms FILO (or vice versa) we characterise each instance by more than 100 features. These features have been proposed in previous work (for example [21, 25]) to characterise VRP and travelling salesman problem (TSP) instances.

After pre-processing the raw features we propose mixture discriminant analysis (MDA) [14] to perform supervised dimension reduction and classification. MDA (which has not been previously used in ISA) is an extension of linear discriminant analysis (LDA). Both LDA and MDA implicitly project the data onto a lower dimensional subspace and perform classification based on these projections. LDA models each class as a Gaussian distribution with a covariance matrix that is common across classes. If there are K classes the LDA classifier discriminates among them using linear separators (boundaries) in a space of at most $K - 1$ dimensions. In our case where $K = 2$ this means that LDA projects the data

onto a vector (line), and then predicts the class by comparing the one-dimensional projection to a scalar threshold. This is too restrictive in our case and the resulting classifier is not competitive. In MDA, each class is modelled as a mixture of Gaussians (again with a common covariance matrix). Compared to LDA, it is a much more flexible model both in terms of the shape of class separators and in terms of the number of dimensions used for classification. This allows us to generate more than one features that are maximally useful for discrimination and interpretation. A key advantage of supervised dimension reduction methods like LDA and MDA is that the lower dimensional representation is optimal for the corresponding classification model. In other words, dimension reduction and classification are treated under a common framework, rather than as two separate steps. In addition to being novel in ISA, MDA is a lot simpler than more sophisticated classifiers like support vector machines (SVMs) which have previously been used.

A central finding from the ISA process is that performance differences between FILO and HGS are small and that which algorithm is the best depends crucially on the allocated time budget. Overall, FILO performs better for more instances when the time budget is short, while the opposite holds when more time is allocated. In either case there is not one algorithm that consistently outperforms the other, and this is reflected in the algorithm footprints. The MDA classifiers developed also allowed us to obtain key insights into the characteristics of problem instances that determine the best algorithm both for short and longer time budgets.

The rest of the paper is organised as follows. In Section 2, we summarise the ISA methodology, also briefly describing MDA. We then describe the CVRP and the metadata used for the analysis in Section 3, followed by the description of the novel instance generator in Section 4. Finally, we present the results of the ISA in Section 5, followed by the conclusions in Section 6.

2 Instance Space Analysis

The algorithm selection problem (ASP) [26] involves a problem space, $\mathcal{P}$, that contains a set of instances of a problem (in our case the CVRP); and an algorithm space, $\mathcal{A}$, which consists of K algorithms to solve the instances in $\mathcal{P}$. The aim is to predict which algorithm in $\mathcal{A}$ will achieve the best performance $y \in \mathcal{Y}$ for a given problem instance $x \in \mathcal{P}$. ASP is therefore a classification problem.

Instance space analysis (ISA) [33] aims to achieve a deeper understanding of the relationship between problem instances $x \in \mathcal{P}$, and algorithm performance y, by providing a framework to evaluate the strengths and weaknesses of algorithms. Key features of ISA is the visualisation of problem instances and the measurement of an algorithm's footprint. The latter is defined as the region in the instance space where good or best performance is expected from this algorithm. Elements of the ISA methodology have been applied to many combinatorial optimisation problems [3, 29, 31], while the methodology has been described in full by Smith-Miles and Muñoz [33]. In this section we summarise the ISA steps, and describe the differences in approach we took in this study.

First, the features and performance metrics are preprocessed. A binary performance metric, $\mathbf{Y_{bin}}$, is calculated from $\mathbf{Y}$ based on a user-defined threshold for "good" performance. This metric can either be absolute (compared to a fixed threshold), or relative (compared to the performance of the best algorithm for a given instance).

The feature selection method, abbreviated as SIFTED [33], aims to identify a set of features that are most strongly correlated with algorithm performance, and are uncorrelated with each other. SIFTED applies k-means clustering to detect groups of similar features. To determine the number of clusters we considered different values of k and chose the one that produced the highest average silhouette coefficient.

The next step is to reduce the feature space to two-dimensions, which are then used for both visualisation and classification. We considered several approaches for dimension reduction and classification, including the combination of PILOT [19] and support vector machines (SVMs) (as suggested in [33]), principal components analysis (PCA) with SVMs, as well as linear discriminant analysis (LDA), and MDA. Our results suggest that LDA classifiers underfit the data, but MDA achieves as good performance as more sophisticated methods like SVMs which require tuning of the type of kernel and its parameters. Given the inherent dimension reduction step in MDA we chose this method, and all the results presented in this paper are based on this classifier. Since, MDA has not been previously used in ISA we provide a brief description of this classifier in the next subsection.

2.1 Linear and Mixture Discriminant Analysis

LDA assumes that observations from each class are sampled from a Gaussian distribution $N(\mu_k, \Sigma_w)$, where μ_k is different for each class, $k = 1, \ldots, K$, but the covariance matrix, Σ_w, is common across classes. The LDA classification rule results from applying Bayes's rule to estimate the most likely class for an observation. It is easily shown [13, Ch. 4] that the common covariance assumption causes the classes to be separated by a linear boundary.

In the case of K classes, LDA classification is equivalent to nearest neighbour classification in an appropriately defined linear subspace of dimension $K - 1$. This subspace is defined by $K - 1$ most discriminative directions (vectors). This means that LDA performs implicitly dimension reduction. It is also important to note that the dimensionality of the subspace does not depend on the dimensionality of the input space, but only on the number of classes. Moreover, it is also possible to reduce the number of dimensions further, which acts as regularisation to prevent overfitting. This classifier is known as reduced-rank LDA.

In our case we have only two classes, and thus LDA projects the data onto a single vector (line). A linear boundary on a line is effectively a scalar threshold, t, and the LDA classifier in this case reduces to comparing the one-dimensional projection of an observation with the value of the threshold t. Such a classifier is too simple to achieve good separation between the classes in our problem.

MDA, proposed by Hastie and Tibshirani [14], is an extension of LDA which can accommodate problems where a single Gaussian distribution per class is not sufficient, and (or) when linear separators cannot adequately discriminate between the classes. In MDA observations from each class are sampled from a mixture of Gaussian distributions. Specifically, the k-th class is modelled as a mixture of R_k components (sub-classes). Each sub-class is a Gaussian with a

distinct mean vector, $\mu_{k,1}, \ldots, \mu_{k,R_k}$ but the covariance matrix, Σ_w, is common across classes and sub-classes (as in LDA). Allowing for more than one Gaussian components allows MDA to be a much more flexible classifier. Hastie and Tibshirani [14] also presented an extension of MDA to accommodate a reduced-rank solution via optimal scoring. We use the reduced-rank MDA to restrict the projected data onto a two-dimensional subspace.

2.2 Algorithm Recommendation

Following the ISA methodology, an MDA classifier is trained for each algorithm in $\mathcal{A}$, which in our case consists of HGS and FILO only. The aim of this classifier is to predict whether the corresponding metaheuristic exhibits good performance on a given problem instance. Algorithm recommendation (also called the *selector* in ISA) takes as input the binary prediction for the performance of each algorithm, on a given problem instance. If for a given problem instance, there is only one algorithm with (predicted) good performance, then this is selected as the best algorithm. If multiple algorithms are predicted to perform well, then the algorithm whose classifier has the highest cross-validation precision is selected as the best. That is the selected algorithm is the one whose classifier is most reliable. If none of the algorithms are predicted to have good performance, then the algorithm with the highest average performance is selected.

3 Problem Setting

Vehicle Routing Problems (VRPs) represent a category of combinatorial optimisation problems that aim to discover the most efficient route(s) for one or more vehicles. As far as we are aware, the concept was first introduced in 1959 by Dantzig and Ramser [9]. A standard VRP involves a fleet of identical vehicles stationed at a depot and a group of customers requiring a particular service, such as delivery or collection. Each customer must be served exactly once by a single vehicle. The primary objective is generally to minimise the overall travel cost, measured in terms of time, or distance.

Most VRPs can be seen as a generalisation of the well-studied *Travelling Salesman Problem* (TSP). The TSP involves a set of points within a metric space, and the objective is to find the shortest closed path that visits each point exactly once. There is also a significant connection between VRPs and *graph theory*. Specifically, depots and customers are typically represented as the nodes of a complete graph, with edges representing transport links. These edges are often assigned a 'weight' that corresponds to the cost, (in terms of time, or distance) of travelling between the associated pair of nodes. From a graph-theoretic perspective, the TSP is equivalent to finding a minimum-weight Hamiltonian circuit in a complete graph. These insights are valuable for our model, as they enable us to extract features of the problem.

The focus of this study is on a specific variant of the VRP known as the *Capacitated Vehicle Routing Problem* (CVRP). The CVRP is defined as follows: Let $V = \{0, 1, \ldots, n\}$ represent the set of vertices, where vertex 0 corresponds to the depot, and vertices $\{1, \ldots, n\}$ correspond to the customers. Let $E = \{(i, j) \mid i, j \in V, i \neq j\}$ denote the set of edges representing possible travel routes between nodes. Each vehicle in the fleet is located at the depot and has a fixed capacity $Q > 0$. Each customer $i \in V \setminus \{0\}$ has an associated demand $d_i \geq 0$, and the total demand served by any vehicle must not exceed its capacity Q. The problem is to design a set of routes, one for each vehicle, such that each route starts and ends at the depot (0), each customer $i \in V \setminus \{0\}$ is visited exactly once by exactly one vehicle, and the total demand on any route does not exceed the vehicle's capacity Q. The objective is to minimise the total travel cost, defined by a weight $c_{ij} \geq 0$ for each edge $(i, j) \in E$, representing the distance, time, or cost of travelling between i and j. The CVRP is an $\mathcal{NP}$-hard combinatorial optimisation problem, combining elements of the TSP and the Bin Packing Problem. There is a large amount of research on the CVRP, with many survey papers available (e.g., [10]).

3.1 Problem Space

For this study, the initial problem space consists of 1,242 CVRP instances. To ensure consistency with the algorithms under consideration, we focus on mid-sized instances (those with 100 to 1,000 customers) since these algorithms were configured and tested on instances within this range. Our problem set includes 30 instances from six well-established sources [5, 6, 11, 4, 27, 12], which have been widely used in the literature for developing both exact and heuristic algorithms. Additionally, we include 12 instances adapted from real-world routing problem instances that featured in the 2021 DIMACS challenge [22].

Each of these relatively small sets has several limitations including a lack of variety in characteristics, and an unrealistic or artificial construction. These limitations were among the reasons why Uchoa et al. [35] proposed a method for generating new CVRP instances and published a set of 100 instances, referred to as **Set X**. Since then, Set X has been frequently used for testing and developing algorithms. The generator, implemented by Queiroga et al. [23], includes parameters to determine the number of nodes, depot position, customer location distribution, demand distribution and vehicle capacity. In terms of customer locations, there are three options: (i) customers are randomly placed on the grid; (ii) a set of S customers are randomly placed on the grid, with the remaining customers clustered around those points; and (iii) half the customers are clustered using the previous scheme, while the other half are randomly positioned.

We introduce a fourth option for generating customer locations, which follows an exponential distribution. Randall [24] propose a modification of a population density model: $f(d) = \lambda \exp(-\gamma d)$, where $f(d)$ is the population density function, d is the distance from the city centre, λ represents the population density at the city centre, and γ is the density gradient. Following this work, we assume the city centre is at the centre of the instance; set the population density to zero for points outside the city limits circle; and set $\gamma = \ln(10) \approx 2.303$, which means that the density at the centre of the circle is ten times the density at the boundary of the circle. In addition to Set X, we included a further 1100 instances using their generator.

3.2 Feature Space

We construct a set of 105 features across six groups to characterise the CVRP instances. Many of these features have previously been used to characterise TSP and VRP instances [17, 21, 25, 28, 34].

Customer demand: We summarise the demand distribution normalised by the vehicle capacity (assuming we are only considering instances with a homogeneous fleet). We also include vehicle capacity and minimum fleet size.

Distance matrix: We calculate pairwise Euclidean distances between all customers. We also normalise the coordinates of each instance to a 1000×1000 grid in order to make the problem instances more comparable. We use both the normalised and original distance matrix when calculating these features. The coordinates of, and distances to the depot and the instance centroid are also considered.

Nearest neighbour distance: For each node in the instance, we calculate the Euclidean distance to its nearest neighbour. The statistics of nearest neighbour distances describe the uniformity of the problem instance.

Minimum spanning tree (MST): The MST of a graph is a subset of the edges that connect all the vertices without any cycles and have the smallest possible total edge weight. We construct the MST for each instance using the Kruskal algorithm [16]. We use the summary statistics of edge distances in this MST. The node degrees of this MST also show the connectivity of the graph.

Geometric: Geometric features are based on the convex hull of the problem instance and reflect the spread of the nodes in the plane. Simple geometric features are the perimeter and area of the rectangle containing the nodes.

Clustering: The customers in a problem instance are clustered using the DBSCAN algorithm [38]. We use features related to the distances and the demand, within and across the clusters. We also include in our set of features the number of clusters, the number of clusters relative to the number of cities and the number of vehicles.

The full list of these features are included in the supplementary materials.

3.3 Algorithm Space

We consider FILO [1] and HGS [37], two state-of-the art open-source metaheuristic solvers for the CVRP. There are other open-source metaheuristic solvers, such as those implemented in the Google OR tools libraries [20], however these are significantly outperformed by FILO and HGS across the entire problem space.

FILO – short for Fast ILS Localized Optimization – is an iterated local search algorithm, with a simulated annealing-based neighbour acceptance criterion [1]. In FILO, an initial infeasible solution is constructed using a variant of the savings algorithm [7], followed by a route minimisation procedure. This solution is then further improved by the core optimisation procedure which includes ruin-and-recreate and local search heuristics.

HGS – a hybrid genetic search algorithm – uses both crossover- and neighbourhood-based search heuristics [37]. After initialising a population, new solutions are iteratively generated by selecting and recombining two parents, then improving this new solution with a local search, and inserting the resulting candidate solution into the population.

3.4 Performance Space

A time budget can be set as a stopping criterion for both FILO and HGS. We measure performance at the end of two different fixed budgets: 120 seconds; and 120 seconds per 100 customers. We will refer to these as **short** and **full** budget, respectively. The performance analysis in [37] was based on runs of the algorithms using a stopping criterion similar to the full budget considered here. For each budget we record the **cost** which corresponds to the total distance of all the routes in the solution. Each algorithm is run 20 times and the median cost is used as our performance metric.

4 CVRP Instance Generator

For ISA insights to be valuable, a diverse set of problem instances is required [32]. We propose a method to generate new CVRP instances by modifying existing problem instances. In this approach, we first cluster the nodes (customers) in a given instance. Then nodes that would be considered "outliers" based on the identified clusters are added. Specifically we apply DBSCAN [38] to the nodes of a given instance to find clusters and outliers. For this algorithm, we calculate the maximum distance between points for them to be considered in the same cluster by assuming uniform distances between nodes, and the minimum number of points to be considered a cluster to 4, as done by Steinhaus [34].

For each cluster found:

(1) Calculate the distance between cluster centroid and depot, t_i.
(2) Find a ***synthetic outlier*** for each cluster along the direction of the line between the depot and cluster centroid with distance $m \cdot t_i$ (where m is a parameter to be set).
(3) If we aim to maintain the instance size, the node closest to the cluster centroid is replaced by the synthetic outlier. The new node is given the demand of the node it is replacing.
(4) Otherwise, we add the synthetic outlier to the problem instance and assign it the median demand for this cluster.

For each instance, the number of new customers is equal to the number of clusters found. Some problem instances - such as those with nodes all equidistant from each other - will not result in a modified instance. This occurs when DBSCAN finds no clusters.

The choice of m affects the structure of the new instance: If $m = 0.5$, all the new customers are closer to the depot than the other points in the existing clusters. If $m = 1$, the new customers will be at the cluster centroids and are not "outliers". If $m = 2$, all the new customers will be further from the depot than the existing customers, with at least some of the new customers becoming the furthest point from the depot in the instance. Figure 1 illustrates this procedure for $m = 0.5$ and $m = 2$.

There are two types of problem instances in the current problem space that we are interested in modifying. First are those which violate the assumption that an algorithm is likely to perform in the same way on instances with *similar* features [3]. For simplicity we say instances who have a different class label from its nearest neighbour violate this assumption. Second are instances in sparse areas. We define these as instances that have fewer than $c = 3$ of its nearest neighbours within a radius of θ.

We then generate new instances by modifying the instances that meet either of those criteria. In order to not add new instances to already dense areas and also to save resources needed to generate

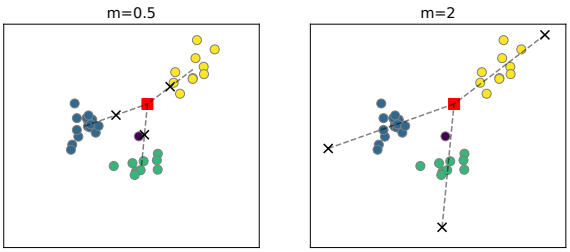


Figure 1: Illustration of instance generator. DBSCAN identifies clusters of customers, a new customer is placed in the direction of each of the (dashed) lines between the cluster centroids and the depot (square). On the right, when $m = 2$, the line is extended past the cluster centroid for the new customer.

additional performance data, we only keep modified instances that are further than θ from the original instance.

The Set X generator [23] we described in Section 3.1 is capable of creating problem instances with a specified clustering structure. In the literature, the clustering structure of nodes has been analysed and used to more efficiently solve different routing problems [8, 39]. Here, we modify this structure (whether specified directly or not) to generate instances that are similar enough to the original problem instances to be relevant.

5 Experimental Analysis

The performance data was generated using the C++ implementations of HGS [36] and FILO [2] on an Intel(R) Xeon(R) CPU at 2.60GHz. ISA was conducted using [15] for MDA and [18] for preprocessing and constructing footprints [1].

The best algorithm for a given problem instance depends on the allocated time budget. In the initial problem space, when the short time budget is used FILO outperforms HGS in 55% of the considered instances, while when the full time budget is allocated HGS outperforms FILO on 52% of the instances. We need to clarify that in approximately 10% of the instances HGS and FILO have equal median cost. We break such ties by setting the best algorithm to be the one that identified the lowest cost solution in the shortest time. Excluding ties in the case of the full budget, the relative performance difference between HGS and FILO performance is on average 0.2%, but it can reach up to 1.8%. For the short budget the average performance difference is 0.6%, and the maximum is 4.3%.

We first conducted a preliminary ISA using the 1,242 problem instances detailed in Section 3.1. Details of the constructed instance space are included in the supplementary materials. Using the lower-dimensional projection of the initial instance spaces, we identified problem instances that are located in sparse areas, and those that have different algorithm performance compared to their nearest neighbours. We expanded the problem set further by modifying these problem instances. Table 1 reports the number of original instances that were selected to be modified, and the size of the resulting problem sets.

[1]The metadata and code for this analysis are included in the repository at https://github.com/danotice/CVRP-ISA-with-MDA

Table 1: Numbers of original instances selected to be modified, modified instances included, instances in expanded problem set in total, and proportion of the total number instances where HGS has better median performance.

Budget	Selected Initial	New Modified	Total Number	HGS Proportion
Short	245	576	1818	0.48
Full	236	696	1938	0.63

Table 2 lists all the selected features for each of the analyses using the SIFTED algorithm [33] which is based on binary performance. Several of the selected features and apparent relationship with algorithm performances are maintained as we expand the instance space. This is positive as it indicates that the new instances enrich the instance space without reversing previously detected patterns. Table 2 shows that the number of customers and the proportion of distinct distances are useful for distinguishing good performance for FILO and HGS. For the full budget models, the mean degree of the MST is selected, which has a direct exponential relationship with the number of customers. Also useful are features relating to the demand distribution and vehicle capacity, as well as customer distances from the depot.

Table 2: Selected features by SIFTED for each model setting. Features that are selected both in initial and expanded problem space are in bold.

Initial Space	
Short	**n**, distance from depot (range), demand (std), **demand to capacity**, **customers to vehicle**, **points on convex hull (prop)**
Full	distance from depot (max), demand (std), demand (min), **customers to vehicle**, **MST degree (mean)**, **points on convex hull (prop)**
Expanded Space	
Short	**n**, distinct distances, distances of kNN of depot (std), demand (min), **demand to capacity**, **customers to vehicle**, **points on convex hull (prop)**, cluster count
Full	distinct distances, distance from depot (range), distances of kNN of depot (std), demand (mean), demand to capacity, **customers to vehicle**, **MST degree (mean), points on convex hull (prop)**

In addition to the features automatically selected for the expanded instance spaces, we include the coefficient of variation (CV) of the distances between nodes in clusters and the cluster centroid. We include this feature because the new instance generator modifies the clustering structure of existing problem instances. In Figure 2 we see that the original instances have a higher cluster distance CV than the modified instances, but there is no distinction in locations between original and modified instances in the projected spaces.

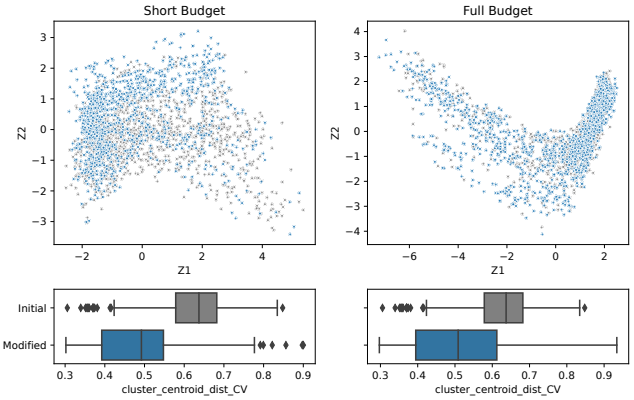


Figure 2: (Top) Two dimensional projections of expanded instance spaces showing the locations of the initial and modified problem instances. (Bottom) Boxplots of the CV of the distances between nodes in clusters and the cluster centroid, group by problem instance type.

The algorithms perform similarly for most of the modified instances added to the problem space when compared to the original problem instance. The proportion of modified instances with the same best algorithm as the corresponding original instance is 0.78 and 0.83 for the short and full budgets respectively. Most of the modified instances for which this is not true are located near the edges of the algorithm's footprints (which we will describe later). This is further indication of how the new instances enrich the instance space – particularly those that were added to regions in the initial instance space with conflicting performance of algorithms.

Because of the proportion of instances with tied performance, we used the **relative binary performance threshold of zero** – meaning, an algorithm is said to have "good" performance for an instance when it has median cost equal to the median cost of the better algorithm. As a result, we train an MDA classifier to predict if FILO is at least as good as HGS, and another for the reverse. The Selector then combines these predictions to recommend an algorithm for each instance.

In the setting which uses the full budget median cost, shown in Figure 3, HGS is selected as the better algorithm for all the instances where the HGS binary classifier predicts good performance. The classifier achieves 84% accuracy and 92% precision on an additional set of 500 modified instances we generated to be test data. The accuracy for the Selector is 83%, which is higher than the accuracy of the binary classifier for FILO, which is 76%. For the short budget setting, the pattern of selection is the opposite, with FILO being selected as the better algorithm for all the instances where the FILO binary classifier predicts good performance. In both settings the Selectors have accuracy and precision of at least 80%.

In addition to predictive models, we also construct **algorithm footprints** for each of the algorithms using the low-dimensional representations of the selected features and the observed algorithm performance (see Figure 4). Footprints are constructed based on high density clusters of instances with good/best performance [33]. These are useful to visualise areas of good and best performance, and we are able to calculate metrics for objective evaluation of these visualisations. Table 3 shows normalised area and density, as well as the purity of the good and best footprints for each algorithm.

In the expanded instance space for the full budget runs, HGS outperforms FILO for 63% of the instances (see Table 1). As a result, HGS has larger but less dense footprints than FILO. This is particularly true for the best performance footprints, where FILO footprint is more than 6 times as dense as the entire instance space on average, but covers only 4% of the total area (see Table 3). This shows that the instances for which we expect FILO to be the better algorithm have very similar characteristics. We observe the reverse (to a lesser extent) for the short budget runs where FILO outperforms HGS for 52% of the instances. We are able to describe the locations of these denser best footprints using the selected features.

Considering the **relationship between features and algorithm performance**, the following observations are valid irrespective of budget:

- HGS has better or equal performance to FILO for instances with more than 6% of the customers on the convex hull of the problem instance, percentage of distinct distances more than 0.012%, or a total demand to capacity ratio of less than 0.9.
- FILO has better or equal performance to HGS for instances with the following combination of features - total demand to capacity ratio greater than 0.95 and more than 650 customers; or a customer to vehicle ratio of greater than 40 and fewer than 300 customers.

Figure 5 shows several of the relevant features in the projected space.

While there are similar relationships when considering different budgets, **performance regret** changes. Performance regret is defined as the relative additional cost incurred by selecting a sub-optimal algorithm for a given problem instance. Specifically, the regret of algorithm a on problem instance i is,

$$\text{regret}_{i,a} = \frac{(y_{i,a} - y_i^\star)}{y_i^\star},$$

where $y_i^\star = \min_{\alpha' \in \mathcal{A}}(y_{i,\alpha'})$.

In the expanded spaces, HGS and FILO have tied median cost for 12% and 21% of the instances for the short and full budget runs, respectively. As shown in Figure 6 regret is higher for the short budget runs compared to the full budget runs. While one may consider regret in the full budget case trivial, this is not the case for the short budget. As a result, the cost of misclassification for HGS is higher for the short budget, than the corresponding cost FILO over the full budget, as the difference between HGS's performance and the performance for the better algorithm is greater. We see the value of the prediction models as we are able to correctly predict the better algorithm with reasonable accuracy, and the regret of the misclassified instances is low.

6 Conclusions

In this work we performed instance space analysis for the CVRP to understand when FILO and HGS – two state-of-the-art metaheuristic algorithms – find lower cost solutions. For our analysis, we used mixture discriminant analysis for projecting the features to

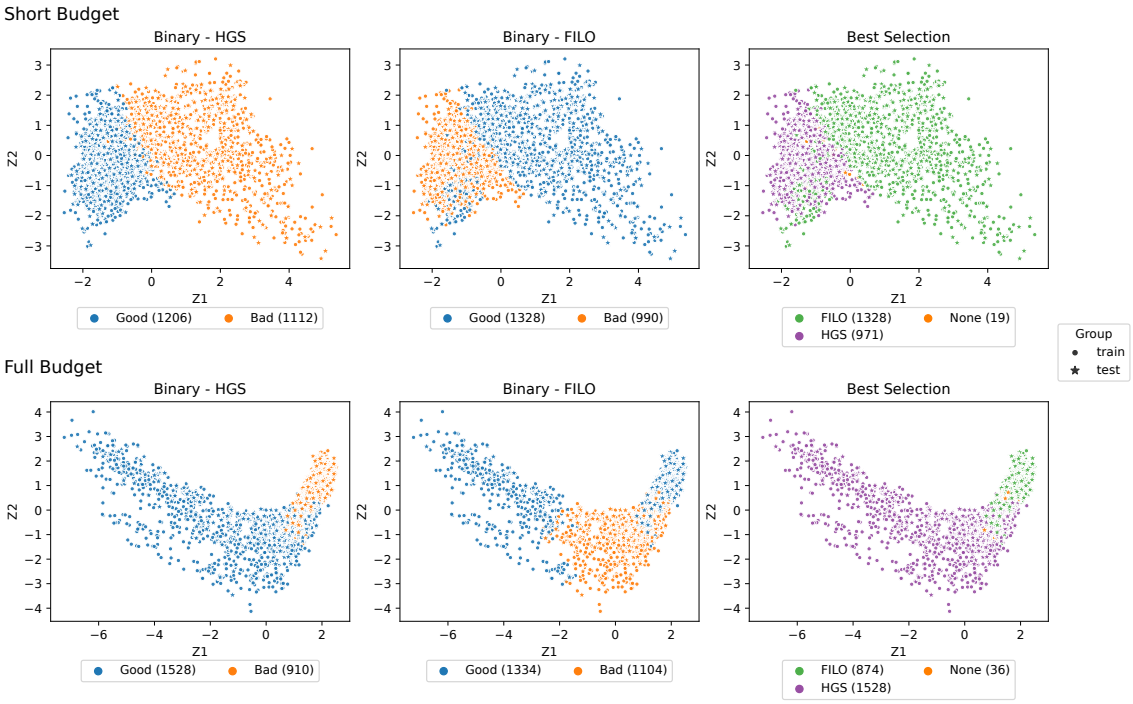


Figure 3: Two dimensional projections and classification for MDA models of good/ bad performance for HGS and FILO as well as the selector. Classifiers for the short budget case appear on the top row and for the full budget in the bottom row.

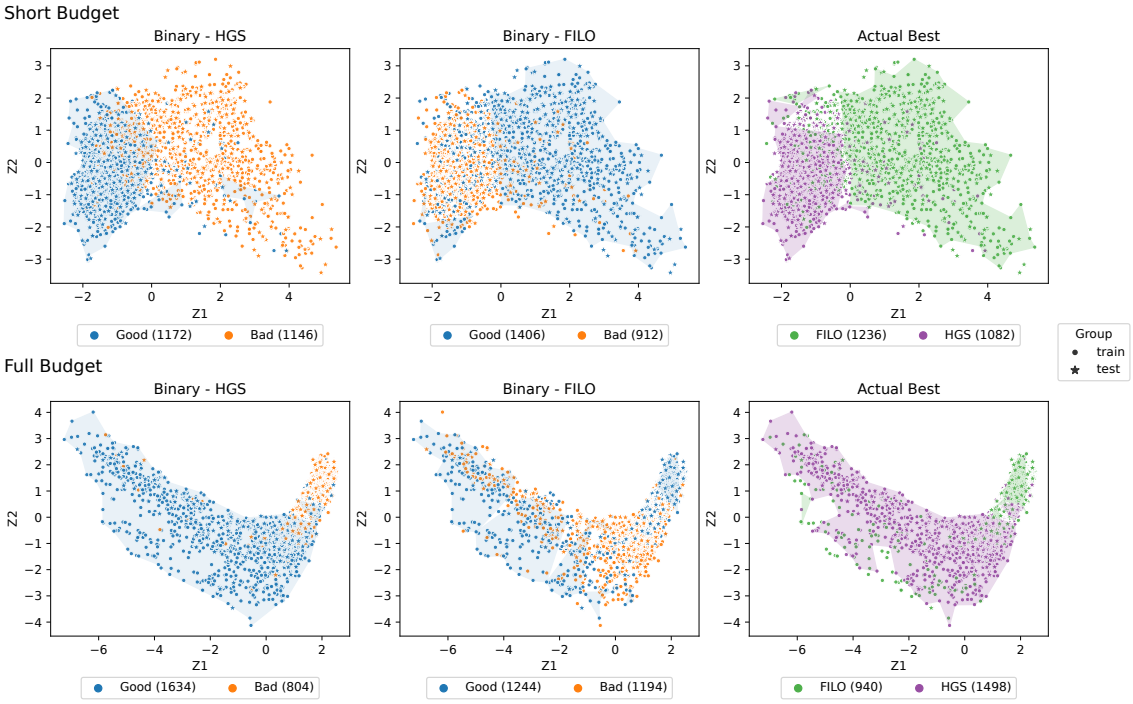


Figure 4: Two dimensional MDA projections and observed binary performance for HGS and FILO, as well as the actual best algorithm. Highlighted are the good/best algorithm footprints. Projections for the short budget case appear on the top row and for the full budget in the bottom row.

Table 3: Footprint evaluation metrics for expanded instance spaces.

		Good			Best		
Budget	Algorithm	Area	Density	Purity	Area	Density	Purity
Short	**HGS**	0.342	1.747	0.820	0.237	2.060	0.812
	FILO	0.698	0.641	0.887	0.654	0.645	0.906
Full	**HGS**	0.959	0.716	0.901	0.818	0.813	0.824
	FILO	0.555	0.765	0.777	0.043	6.151	0.809

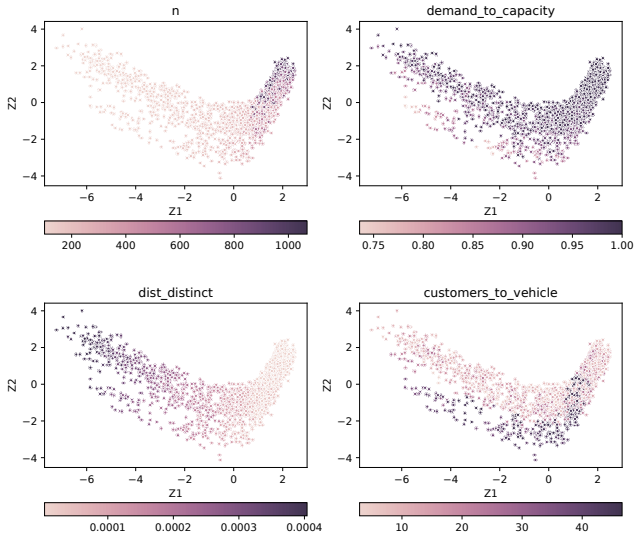


Figure 5: Two dimensional projection of the expanded full budget instance space showing the following relevant features - number of customers, ratio of total demand to total vehicle capacity, proportion of distinct distances, average number of customers per vehicle.

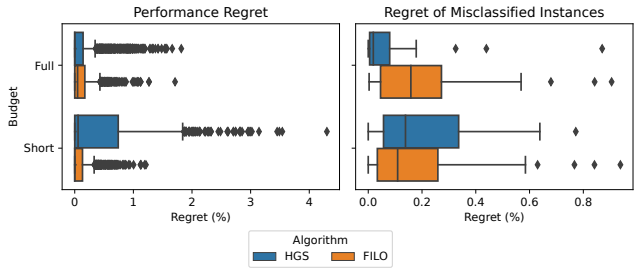


Figure 6: (Left) Boxplots of the performance regret for all instances used to train the models. (Right) Boxplots of performance regret for test instances that were misclassified.

a two-dimensional space and the prediction of binary performance. Using MDA allowed us to simplify the ISA as dimension reduction and classification are treated under a common framework, rather than as two separate steps, without compromising the quality of the predictions or the value of insights.

We constructed an extensive feature set to characterise the CVRP based on previous studies. Using these features we were able to describe regions in the instance space, and therefore corresponding problem instances, where either algorithm is likely to have good performance. Among the features useful for defining these regions of good performance were the number of customers and the ratio of total demand to total capacity.

We also proposed a novel instance generator which we were able to use in our generation of a diverse set of problem instances. The analysis demonstrated that our novel instance generator effectively extends the problem space. We observe how making small changes to the customer distribution often does not change the relative performance of the algorithms. The instance generator is able to modify existing problem instances in a way that is relevant for real-world context, as clusters of customers may be taking into account when solving routing problems. In the future, it is possible to explore the effects of varying the extent to which the customer distribution is, as well as modifying other features that can be measured within the clusters - such as demand distribution.

One of our main aims was to explore how changing the stopping criteria of the algorithms affects the performance. We considered two time budgets and concluded that the relative performance of the algorithms depends on which budget is used. HGS outperforms FILO for more instances when the full budget is used, which is reflected in the size of the footprints. The opposite is true for when the short budget is used. Also more generally, we see that the difference in performance between FILO and HGS on a given instance is larger when the short budget is used than when the full budget is used. Our analysis highlights the nuance in the comparison of FILO and HGS, as both the characteristics of the problem instances and the time allocated to solve the problem play a crucial role in algorithm recommendation.

Acknowledgments

This paper is largely based on work completed while D. Notice was part of the UKRI Engineering and Physical Sciences Research Council funded STOR-i Centre for Doctoral Training (EP/S022252/1). The project was also funded in part by Tesco Stores Limited. H. Soleimani and M.A. Muñoz were funded by the Australian Research Council through grants No.: FL140100012 and IC200100009. H. Soleimani developed the instance generator while working at The University of Melbourne. We thank Kate Smith-Miles for her input on the instance generator.

References

[1] Luca Accorsi and Daniele Vigo. 2021. A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. *Transportation Science*, 55, 4, 832–856. DOI: 10.1287/trsc.2021.1059.

[2] [SW] Luca Accorsi and Daniele Vigo, filo 2021. VCS: https://github.com/acco93/filo.

[3] Hossein Alipour, Mario Andrés Muñoz, and Kate Smith-Miles. 2023. Enhanced instance space analysis for the maximum flow problem. *European Journal of Operational Research*, 304, 2, 411–428. DOI: 10.1016/j.ejor.2022.04.012.

[4] Philippe Augerat, D Naddef, JM Belenguer, E Benavent, A Corberan, and Giovanni Rinaldi. 1995. Computational results with a branch and cut code for the capacitated vehicle routing problem. Report. Université Joseph Fourier.

[5] N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, (Eds.) 1979. *The vehicle routing problem. Combinatorial optimization*. John Wiley and Sons, London.

[6] Nicos Christofides and Samuel Eilon. 1969. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20, 309–318. DOI: 10.1057/jors.1969.75.

[7] G. Clarke and J. W. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, 4, 568–581. DOI: 10.1287/opre.12.4.568.

[8] Joao Guilherme Cavalcanti Costa, Yi Mei, and Mengjie Zhang. 2020. Cluster-based hyper-heuristic for large-scale vehicle routing problem. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. DOI: 10.1109/cec48606.2020.9185831.

[9] G.B. Dantzig and J.H. Ramser. 1959. The truck dispatching problem. *Manag. Sci.*, 6, 80–91. DOI: 10.1287/mnsc.6.1.80.

[10] R. Elshaer and H. Awad. 2020. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Comput. Ind. Eng.*, 140. DOI: 10.1016/j.cie.2019.106242.

[11] Marshall L. Fisher. 1994. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42, 4, 626–642. DOI: 10.1287/opre.42.4.626.

[12] T.G. Crainic and G. Laporte, (Eds.) 1998. *The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. Fleet Management and Logistics*. Springer, Boston, MA. Chap. Chapter 2, 33–56. DOI: 10.1007/978-1-4615-5755-5_2.

[13] T. Hastie, R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning*. Springer New York, (Ed.) (2nd ed.). DOI: 10.1007/978-0-387-84858-7.

[14] Trevor Hastie and Robert Tibshirani. 1996. Discriminant analysis by gaussian mixtures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58, 1, 155–176.

[15] Trevor Hastie and Robert Tibshirani. 2024. *mda: Mixture and Flexible Discriminant Analysis*. R package version 0.5-5. https://CRAN.R-project.org/package=mda.

[16] Joseph B. Kruskal. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7, 1, 48–50. DOI: 10.1090/s0002-9939-1956-0078686-7.

[17] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Markus Wagner, Jakob Bossek, and Frank Neumann. 2013. A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, 69, 2, 151–182. DOI: 10.1007/s10472-013-9341-2.

[18] [SW] M.A. Muñoz and K. Smith-Miles, Instance Space Analysis: A toolkit for the assessment of algorithmic power 2020. DOI: 10.5281/zenodo.4750845.

[19] Mario A. Muñoz, Laura Villanova, Davaatseren Baatar, and Kate Smith-Miles. 2018. Instance spaces for machine learning classification. *Machine Learning*, 107, 1, 109–147. DOI: 10.1007/s10994-017-5629-5.

[20] Laurent Perron and Vincent Furnon. 2023. OR-Tools. Computer Program. (2023). https://developers.google.com/optimization/.

[21] Josef Pihera and Nysret Musliu. 2014. Application of machine learning to algorithm selection for TSP. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, 47–54. DOI: 10.1109/ictai.2014.18.

[22] Eduardo Queiroga. [n. d.] Capacitated vehicle routing problem library. Retrieved June 13, 2023 from http://vrp.galgos.inf.puc-rio.br/index.php/en/.

[23] Eduardo Queiroga, Ruslan Sadykov, Eduardo Uchoa, and Thibaut Vidal. 2022. 10,000 optimal CVRP solutions for testing machine learning based heuristics. In *AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR)*.

[24] Matt Randall. 2024. *Putting the ping into online shopping: Dynamic Heuristics for Vehicle Routing and Slot Offering*. Ph.D. Dissertation. Lancaster University.

[25] J. Rasku, T. Kärkkäinen, and N. Musliu. 2016. Feature extractors for describing vehicle routing problem instances. In *OpenAccess Series in Informatics*. Vol. 50, 7.1–7.13. DOI: 10.4230/OASIcs.SCOR.2016.7.

[26] Morris Rubinoff and Marshall C. Yovits, (Eds.) 1976. *The algorithm selection problem. Advances in Computers*. Vol. 15. Elsevier, 65–118. DOI: 10.1016/S0065-2458(08)60520-3.

[27] Yves Rochat and Éric D. Taillard. 1995. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1, 1, 147–167. DOI: 10.1007/bf02430370.

[28] K. Smith-Miles and J. van Hemert. 2011. Discovering the suitability of optimisation algorithms by learning from evolved instances. *Annals of Mathematics and Artificial Intelligence*, 61, 2, 87–104. DOI: 10.1007/s10472-011-9230-5.

[29] Kate Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis. 2014. Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45, 12–24. DOI: 10.1016/j.cor.2013.11.015.

[30] Kate Smith-Miles and Simon Bowly. 2015. Generating new test instances by evolving in instance space. *Computers & Operations Research*, 63, 102–113. DOI: 10.1016/j.cor.2015.04.022.

[31] Kate Smith-Miles, Jeffrey Christiansen, and Mario Andrés Muñoz. 2021. Revisiting where are the hard knapsack problems? via instance space analysis. *Computers & Operations Research*, 128. DOI: 10.1016/j.cor.2020.105184.

[32] Kate Smith-Miles and Leo Lopes. 2012. Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39, 5, 875–889. DOI: 10.1016/j.cor.2011.07.006.

[33] Kate Smith-Miles and Mario Andrés Muñoz. 2022. Instance space analysis for algorithm testing: methodology and software tools. *ACM Computing Surveys*, 55, 12, 1–31. DOI: 10.1145/3572895.

[34] Meghan Steinhaus. 2015. The application of the self organizing map to the vehicle routing problem. *Open Access Dissertations*. DOI: 10.23860/diss-steinhaus-meghan-2015.

[35] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. 2017. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257, 3, 845–858. DOI: 10.1016/j.ejor.2016.08.012.

[36] [SW] Thibaut Vidal, HGS-CVRP version 2.0.0, 2022. VCS: https://github.com/vidalt/HGS-CVRP.

[37] Thibaut Vidal. 2022. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140. DOI: 10.1016/j.cor.2021.105643.

[38] Xu Xiaowei, M. Ester, H. P. Kriegel, and J. Sander. 1998. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings 14th International Conference on Data Engineering*, 324–331. DOI: 10.1109/icde.1998.655795.

[39] M. S. Zaeri, J. Shahrabi, M. Pariazar, and A. Morabbi. 2007. A combined spatial cluster analysis - traveling salesman problem approach in location-routing problem: a case study in iran. In *2007 IEEE International Conference on Industrial Engineering and Engineering Management*, 1599–1602. DOI: 10.1109/ieem.2007.4419462.