A Feature-Based Comparison of Local Search and the Christofides Algorithm for the Travelling Salesperson Problem

Samadhi Nallaperuma, Markus Wagner, Frank Neumann
Evolutionary Computation Group
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia

Bernd Bischl, Olaf Mersmann, Heike Trautmann
Statistics Faculty
TU Dortmund University
44221 Dortmund, Germany

ABSTRACT

Understanding the behaviour of well-known algorithms for classical NP-hard optimisation problems is still a difficult task. With this paper, we contribute to this research direction and carry out a feature based comparison of local search and the well-known Christofides approximation algorithm for the Traveling Salesperson Problem. We use an evolutionary algorithm approach to construct easy and hard instances for the Christofides algorithm, where we measure hardness in terms of approximation ratio. Our results point out important features and lead to hard and easy instances for this famous algorithm. Furthermore, our cross-comparison gives new insights on the complementary benefits of the different approaches.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, Algorithms, Performance

Keywords

Traveling Salesperson Problem, Approximation Algorithms, Local Search, Classification, Prediction, Feature Selection

1. INTRODUCTION

Our goal is to understand the performance of algorithms for hard optimisation problems such as the Travelling Salesperson Problem (TSP). This understanding is essential for algorithm design and automated algorithm selection. In both the artificial intelligence (AI) and operational research communities, this topic has become a major point of interest. Thus, various kinds of attempts have been made theoretically and empirically. Classical approaches taking a worst-case or an average-case perspective hardly capture what is happening for real instances. For a given instance I of a combinatorial optimisation problem, it is often hard to predict the performance of an algorithm A without running A on I.

Hyper heuristics in the optimisation domain and meta-learning in the machine learning domain focus on finding the conditions that determine algorithm performance in advance. Smith-Miles and Lopes [8] classify the research on problem hardness analysis into two different directions. The first direction is to consider the problem as a learning problem, where automatic algorithm selection [3] is done based on learned knowledge from previous algorithm performance. The second direction is to analyse the algorithms and problems theoretically [7, 10, 4] and experimentally [8] [13] to understand the reasons for performance on different problem instances. This understanding is the key to future algorithm design for more complex real world problems.

Our study considers both approaches, where we investigate the performance of important algorithms for the TSP on different instances. Heuristic methods are frequently used to tackle NP-hard combinatorial optimisation problems. Usually, they do not provide any performance guarantees. In contrast to this, approximation algorithms provide guarantees on the quality of a solution that is achieved by running the approximation algorithm. In this paper, we investigate which features make instances of the TSP hard or easy for the well-known Christofides approximation algorithm. Easy and hard instances for this algorithm are generated by an evolutionary algorithm (EA) presented in [6]. Furthermore, we examine the behaviour of a 2-Opt based local search algorithm on these instances and carry out a comparison to a well-known 2-Approximation algorithm.

Our results provide evidence on the capability of individual or combinations of features to classify instances into hard and easy ones. Some features like distance and minimum spanning tree statistics are more effective for this classification for the Christofides algorithm than others like convex hull or mode features. Combined with the analysis of the feature values of the instances of medium difficulty, an increased understanding of the individual feature influences on the approximation quality is provided. Results of the algorithm comparisons enable the analysis of relative strengths of the algorithms on each others' difficult instances as 2-Opt (the Christofides algorithm respectively) outperformed the

Christofides algorithm (2-Opt respectively) on its hard instances. These insights can be used to improve automatic algorithm selection and algorithm design.

The rest of the paper is organised as follows. In Section 2, we introduce the considered algorithms and the approach of selecting features that measure problem difficulty. In Section 3, we carry out the analysis of easy and hard instances for the Christofides algorithm. In Section 4, we compare the performance of different algorithms on their respective easy and hard instances. Finally, we conclude with some remarks.

2. PRELIMINARIES

The Travelling Salesperson Problem (TSP) is one of the most famous NP-hard combinatorial optimization problems. Given a set of n cities $\{1,\dots,n\}$ and a distance matrix $d = (d_{i,j})$, $1 \le i,j \le n$, the goal is to compute a tour of minimal length that visits each city exactly once and returns to the origin. A tour that visits each city exactly once and return to the origin is frequently called a Hamiltonian cycle. Hamiltonian cycles for complete graphs can be represented as permutations of the n cities. For a given permutation $\pi = (x_1,\dots,x_n)$ we denote by

$$c(\pi) = d_{x_n,x_1} + \sum_{i=1}^{n-1} d_{x_i,x_{i+1}}$$

the cost of the tour π.

A wide range of algorithms have been developed for the TSP including approximation algorithms and various heuristic approaches. The approximation ratio of an algorithm A for a given instance I is defined as

$$\alpha_A(I) = A(I)/OPT(I)$$

where $A(I)$ is the tour length produced by algorithm A for the given instance I, and $OPT(I)$ is the length of the shortest Hamiltonian cycle in I. An algorithm A is an r-approximation algorithm if for any valid input I, $\alpha_A(I) \le r$ holds, i. e. the worst case instance can have an approximation ratio of at most r.

Different approximation algorithms have been developed for the TSP. We refer the reader to the book of Vazirani [12] for a comprehensive presentation. In general, the TSP is not only NP-hard but also hard to approximate. We will restrict ourselves to a subset of all TSPs, the class of Metric TSPs. Here the distances between the cities have to fulfill the triangle inequality

$$\forall i,j,k \in \{1,\dots n\}\colon d_{ik} \le d_{ij} + d_{jk}.$$

One of the most prominent approximation algorithms for the Metric TSP is the Christofides algorithm (see Algorithm 1), which achieves an approximation ratio of 3/2. It starts by computing a minimum spanning tree T for the given input. Furthermore, a minimum weight matching M is computed on nodes that have odd degree in T. The graph obtained by combining the edges of T and M is used to compute an Euler tour which is then turned into a Hamiltonian cycle by using short-cuts.

A prominent special case of the Metric TSP is the Euclidean TSP. Here, cities are represented by points in the plane and distances are given by the Euclidean instances between these points. The Euclidean TSP is often considered in experimental investigations. Note, that the Euclidean TSP is still NP-hard but admits a PTAS [2]. However, this algorithm is not

Algorithm 1: Christofides 3/2-approximation algorithm

```
input  : Graph $G$
output: Hamiltonian cycle $\pi$
1 Compute a minimum spanning tree MST $T$ of $G$.;
2 Find a minimum-weight perfect matching $M$ on the set
  of nodes of $T$ having an odd degree.;
3 Combine the edges of $M$ and $T$ to form the graph $U$.;
4 Create an Euler cycle $S$ in $U$.;
5 Obtain a Hamiltonian cycle $\pi$ from $S$ by skipping
  already visited nodes.;
6 return $\pi$;
```

considered to be practical, instead we will investigate the performance of the Christofides algorithm on Euclidean instances of the TSP in greater detail by analyzing features of easy and hard instances.

In practice, heuristic methods such as local search are frequently used to solve instances of the TSP. Our goal is to compare the Christofides algorithm to a standard local search algorithm based on the well-known 2-Opt operator. The complete local search algorithm is given Algorithm 2. It repeatedly checks whether the swapping of two edges in a tour results in a shorter tour. If no improvement can be found any more, the tour is called "2-Optimal" and the algorithm terminates. Note that in [6] a variant is used in which randomness is only induced by varying the initial tour, whereas the 2-opt algorithm is deterministic in always choosing the edge replacement resulting in the highest reduction of the current tour length.

2.1 Hard and easy instance generation

The most generic way to generate hard or easy instances is based on a feature set that is considered to determine problem hardness [9]. Hard or easy instances are generated by setting the values of these features to modify the problem difficulty level. Then algorithm performance is measured on these instances. Smith-Miles and Lopes [9] criticise this conventional approach. The two major drawbacks are the difficulty of generating diverse random instances and the restrictedness of randomly generated benchmark datasets in the spectrum of difficulty. Van Hemert [11] has proposed a different approach. His approach is based on an evolutionary algorithm that evolves instances based on the performance of the algorithm being investigated. After this study on the Lin-Kernighan algorithm [5], there were several more studies that used this approach to generate hard and easy instances for problems like the TSP [8, 6, 9]. Using an evolutionary algorithm, it is possible to evolve sets of extremely hard and easy instances by maximizing or minimizing the fitness (tour length) of each instance. This is essential to achieve diversity within the data set [9]. Therefore, we assume that an evolutionary algorithm based approach can generate a diverse set of easy and hard instances for approximation algorithms as well.

We measure the hardness of an instance I for a given algorithm A by the approximation ratio $\alpha_A(I)$. We drop the subscript A and write $\alpha(I)$ if it is clear which algorithm A is under investigation. Since we only consider deterministic approximation algorithms in this paper, we can obtain $A(I)$ by a single run of algorithm A on a given instance I. However, within the instance generation for 2-opt in [6] it is accounted for randomness by using several different ini-

Algorithm 2: 2-Opt algorithm

```
input  : Graph $G$
output: Hamiltonian cycle $\pi$
1  Choose a $\pi$;
2  while true do
3      best improvement := 0;
4      for i from 0 to number of cities do
5          $x_i$ := $i^{th}$ node in $\pi$;
6          for j from i to number of cities do
7              $x_j$ := $j^{th}$ node in $\pi$;
8              improvement := $distance(\{x_i, x_{i+1}\}) + distance(\{x_j, x_{j+1}\}) - distance(\{x_i, x_j\}) - distance(\{x_{i+1}, x_{j+1}\})$;
9              if improvement > best improvement then
10                 $\pi := \pi'$;
11                 best improvement:=$c(\pi')$;
12                 best current := $i$;
13                 best other := $j$;
14     if best improvment > 0 then
15         comment : swap edges and reverse the cities in between;
16         current:= best current + 1; other := best other;
17         while current <= other do
18             increment current; decrement other;
19             tmp := $\pi[current]$;
20             $\pi[current]$:= $\pi[other]$;
21             $\pi[other]$ := tmp;
22     else
23         break while-loop;
24 return $\pi$;
```

tial tours. $OPT(I)$ is obtained by using the exact TSP solver Concorde [1].

In order to evolve easy and hard instances for approximation algorithms, we use the evolutionary algorithm introduced by Mersmann et al. [6]. The search is guided by the approximation ratio of an instance, which is used as the fitness function in the evolutionary algorithm. It should be noted that this is in contrast to the approach proposed by van Hemert [11] who used algorithm runtime as a measure for the hardness of a particular problem instance. We maximize $\alpha(I)$ in order to generate hard instances and we minimize $\alpha(I)$ in order to generate easy instances for a given fixed algorithm A. For the analysis instance sizes of 25, 50, 100 and 200 nodes are investigated.

Our evolutionary algorithm uses two strategies to create new instances: (1) "local mutation" is performed by adding a small normal perturbation to the location (normalMutation), and (2) "global mutation" is carried out by replacing each coordinate of the city with a new uniform random value (uniformMutation). This later step was performed with a very low probability. The two sequential mutation strategies together enable small local as well as global structural changes in the offspring. The parameters of the evolutionary algorithm are set as follows: *population size* = 30, *generations* = 5000, *time limit* = 24h, *normalMutationRate* = 0.01, *uniformMutationRate* = 0.001, *cells* = 100, and the standard deviation of the normal distribution used in the normal- Mutation step equals *normalMutationSd* = 0.025. The parameter levels were chosen based on initial experiments. For each combination of difficulty and input size, we run the evolutionary process 100 times with different initial populations in order to create a diverse set of hard and easy instances.

2.2 Investigated features

We study features that lead to easy and hard instances in a similar way as Mersmann et al. [6], including statistics based on the distance matrix, the minimum spanning tree, and the convex hull of the cities. The complete set of features is listed in the following.

Distance Features: Features based on summary statistics of the edge cost distribution such as the lowest, highest, mean and median edge costs, the proportion of edges with distances shorter than the mean distance, the fraction of distinct distances, the standard deviation of the distance matrix and the expected tour length for a random tour.

Mode Features: The number of modes of the edge, the cost distribution and related features such as the frequency and quantity of the modes, the mean of the modal values, and the number of modes of the edge cost distribution.

Cluster Features: These features assume that the existence and the number of node clusters relates to algorithm performance. In particular, the number of clusters and mean distances to cluster centroids are determined using different levels of reachability distances of the clustering algorithm GDBSCAN.

Nearest Neighbour Distance Features: Features reflecting the uniformity of an instance such as the minimum, maximum, mean, median, standard deviation and the coefficient of variation of the normalised nearest neighbour distances of each node.

Centroid Features: The coordinates of the instance centroid together with the minimum, mean and maximum distance of the nodes from the centroid.

MST Features: Statistics that are related to the depth and the distances of the minimum spanning tree (MST). These include the minimum, mean, median, maximum and the standard deviation of the depth and distance values of the MST completed by the sum of the distances on the MST, normalised by diving it by the sum of all pair wise distances. This feature group represents the MST heuristic that provides an upper bound for the optimal tour, i.e. the solution of the MST heuristic is within a factor two of the optimal.

Angle Features: Statistics regarding the angles between a node and its two nearest neighbour nodes, i.e. the minimum, mean, median, maximum and the respective standard deviation.

Convex Hull Features: The area of the convex hull of the instance reflecting the spread of the instance in the plane and the fraction of nodes that define the convex hull.

Different TSP instance sizes are considered for the analysis. Cities are generated in $[0, 1]^2$ and placed on a discretised grid enabling cross comparison of features. Instances with varying difficulty levels in between easy and hard are generated by a sophisticated morphing strategy which includes a heuristic point matching strategy between easy and hard instances and computes convex combinations of the respective points of both instance classes. The instances of various difficulty levels will help to increase the understanding of the correlation between instance features, algorithm performance and problem difficulty.

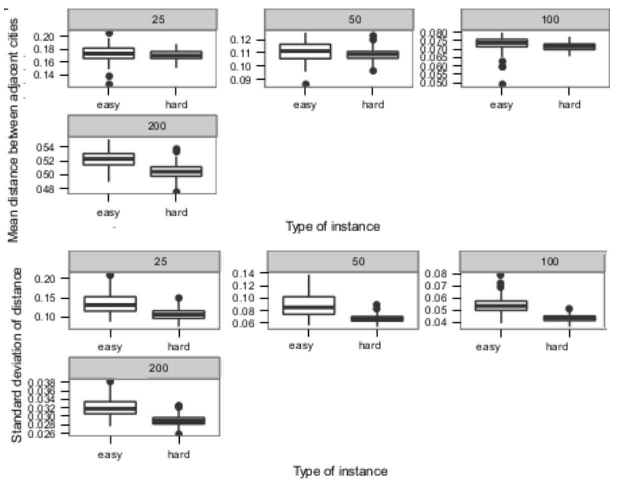


Figure 1: Boxplots of the mean (top) and standard deviations (bottom) of the tour length legs of the optimal tour, both for the evolved easy and hard instances for Christofides.

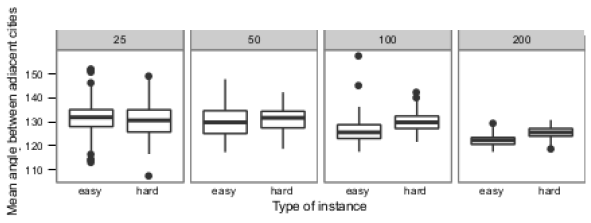


Figure 2: Boxplots of the mean values angle between adjacent cities on the optimal tour for Christofides.

3. ANALYSIS OF THE CHRISTOFIDES ALGORITHM

We now characterize instances of different difficulty for the Christofides algorithm. We start by examining hard and easy instances for the Christofides algorithm. The achieved approximation ratio is close to 1 for all the easy instances and roughly 1.4 for the hard instances. Later on, we morph easy instances into hard ones. The mean distances of the optimal tours of the easy instances are greater than those of the hard instances across all considered instance sizes. Similarly, the standard deviation of the distances of each leg of an optimal tour of the easy instances is considerably higher than for the hard instances. This does not change for increasing instance size (see Figure 1). It is observable that easy instances consist of small clusters of cities as opposed to a more uniform distribution for the hard instances.

As observed in Figure 2, the mean angle of successive tour legs of the easy instances are higher than those of the hard instances (when considering small instances), and lower for larger instance sizes. However, the differences in location are not statistically significant for the small instances. Nevertheless, the results for the larger instance sizes indicate that in this case the instances have higher angles than the easy ones.

Using the boxplots in Figures 1 and 2), we can identify individual features with the capability to differentiate easy from hard instances. This can be refined by using two features to classify instances as easy or hard. Examples for this are shown in Figure 3. There, the top figure shows a combination of distance (standard deviation) and angle (stan-

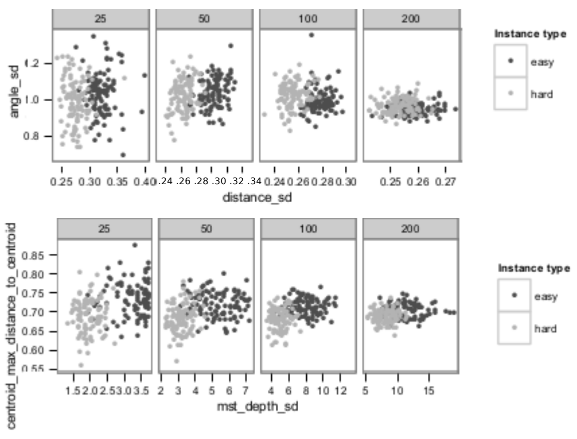


Figure 3: Scatterplots of exemplary feature combinations classifying easy and hard instances for Christofides.

dard deviation) features, while the bottom figure shows a combination of MST (the standard deviation of the depth of MST) and centroid (maximum distance to the centroid) features. As observed in the previous boxplots, the angle features alone do not provide enough information for an accurate classification. Nevertheless, once combined with another features (such as distance), the angle features are capable of discerning easy from hard instances reasonably well. Similarly, two exemplary MST and centroid features are show in the bottom figure, to illustrate that these too can be used to build accurate classification rules.

3.1 Features of instances with different approximation ratios

We create instances with varying difficulty levels by forming convex combinations of easy and hard instances, a process we call morphing in the following. Then, the changes of the feature values with increasing difficulty level are studied in order to understand the influence of the different features on the approximation quality. Figures 9 to 13 show the approximation quality for the Christofides instances of all morphing sequences, for the various instance difficulty levels represented by α. Here, $\alpha \in \{0, 0.2, ..., 0.8, 1\}$ increases from hard to easy instances (left to right). In the following, we will discuss basic observations on the variation of the feature values from hard to easy instances for Christofides, with references to the previous observations on 2-Opt [6].

Figure 9 explains the variation of distance features with the instance difficulty for Christofides. The mean, median and the standard deviation of distances have inverse relationships with the instance difficulty. They decrease drastically when the approximation ratio increases. The maximum distance shows a similar pattern, yet does not have a dramatic change over increasing difficulty. Some features such as the minimum distance and the mean tour length do not exhibit a systematic relationship with increasing α. In most cases the variation of the feature over all values for α is larger for small instances than for larger instances. For example, the standard deviation of the distance for the smallest instance size varies within a range of 0.15 (from 0.25 to 0.4), where as for the largest instance size this range is only 0.05 (from 0.25 to 0.30). This pattern holds even for the features like the mean tour length, minimum and the distinct distance, which do not exhibit a strong relationship with increasing α. Compared to the feature values for 2-Opt [6], some features exhibit a stronger relationship with α for Christofides. For example, the median and the mean show stronger relationships with α for all instance sizes, where 2-Opt had the strong increasing pattern only for the smallest instance size. For the rest of the features, the pattern of this systematic nonlinear relationship is almost similar for both algorithms, in spite of slight differences in exact values of features. In contrast to the distance features, the cluster features shown in Figure 9 do not exhibit a strong systematic relationship with α. The cluster feature values for Christofides show patterns similar to 2-Opt [6], as well as having exact values also much closer to the values of 2-Opt's cluster features.

Figure 10 shows some evidence that the angle features do not have a strong systematic relationship between the feature value and the instance difficulty for Christofides like for 2-Opt [6]. Nevertheless, there is a slight variation observable in the standard deviation, minimum and the maximum angle features. There, feature values decrease (increase in minimum angle accordingly) until the medium difficulty level is reached, and then again increase slightly, over the increasing (decreasing in minimum angle accordingly) α. This provides a hint on the dominance of angle features for the instances with medium difficulty. Thus, we might use the angle features to identify instances with medium difficulty for the approximation algorithm, in spite of its inability to differentiate hard from easy instances. Interestingly, similar observation for the maximum angle feature can be made for 2-Opt as well.

Some of the features in the centroid feature group, like the maximum distance from centroid and the mean distance from centroid, provide a good representation of the instance difficulty, through the systematic nonlinear relationship with increasing feature values over α. However, it is observable that the exact centroid location alone does not provide any insight into the problem difficulty (see Figure 10). The highest range of feature values is observed for the mean distance to centroid in case of the smallest instance size, which is 0.2 (from 0.35 to 0.55). Similar nonlinear relationships are observed for centroid features for 2-Opt [6] also, with slight differences in the range of the feature values, such as a 0.3 to 0.7 range for centroid x for 2-Opt and 0.35 to 0.06 for Christofides.

The convex hull features also reflect an influence on the approximation ratio along α, although this is less prominent than in other feature groups (see Figure 11). Among the two features, only the area of the convex hull relates with instance difficulty, having increasing feature value with α, thus decreasing with the instance difficulty. In terms of the area of the convex hull, Christofides shows patterns inverse to those of 2-Opt [6], where the area grows with the instance difficulty. Moreover, the feature values for the points on the hull also increases over α for 2-Opt, while no such notable variation observed for Christofides.

As shown in Figure 11 mode features do not exhibit any correlation with the instance difficulty, having similar feature values over increasing α. The range of feature value is reduced with the instance size, similar to the convex full feature group. The pattern of variation in mode features for Christofides aligns well with that of 2-Opt [6], except for the mode quantity for larger instance sizes, where 2-Opt has feature values in 0.1 range and Christofides has in 0.01 range.

Strong systematic relationships can be detected for the MST features (see Figure 12). Among this feature group, features representing the depth of the spanning tree have stronger relationship with α than others. These include the depth mean, median, maximum and the depth's standard deviation. Considering the difference in scale, it is not possible to compare the exact values for the variation of feature values among features. However, it is observed visually, that a strong systematic relationship exists for the maximum distance feature, in the case of the smallest instance size. The distance features of the MST show relationships with increasing α, mostly for smaller instance sizes. An interesting observation in this group is that some features are more prominent for smaller instance sizes (distance maximum, distance standard deviation), while some are dominating for larger instance sizes (depth features). Furthermore, it is observed that the minimum statistics of both feature groups, distance and depth (minimum depth and minimum distance) do not indicate any relationship with α. When comparing this with the results for 2-Opt [6], Christofides shows similar patterns of the variation of the MST feature values with α, except for the depth features. There, Christofides shows more consistent relationship of increasing feature value with α for all instance sizes, whereas 2-Opt merely shows a slight variation for the largest instance size. Again, these depth feature values for the largest instance size are decreasing with α, in contrast to the increasing pattern of the Christofides algorithm.

The nearest neighbour distance feature group (see Figure 13) also shows relationships with α. Especially the standard deviation and the coefficient of variation provide stronger relationships than the maximum and the minimum statistics. The coefficient of variation exhibits the largest variation and influence of the feature values. This is observed for the smallest instance size. Instances for both 2-Opt [6] and Christofides algorithms exhibit similar patterns of (nonlinear) relationship of approximation ratio with feature levels, where some features like the median and the mean are decreasing with α, while others like the standard deviation and the coefficient of variation are increasing.

In summary, the variation of feature values over the difficulty level (α) is more prominent in some feature groups than the others. The distance, nearest neighbour, centroid and the MST feature groups have stronger systematic relationships with α than the angle, convex hull, cluster and the mode feature groups. It is observed that some features have greater variation with α than others, even within a single feature group. For example, the area of the convex hull increases drastically with α, while the number of points in the hull stays relatively steady. Interestingly, some features exhibit different tendencies for the smallest and highest instance size, such as the features reflecting the minimum distance to the centroid and mean of angles in Figure 10. This is due to the structural shapes of small and large instance classes. Generally, the standard deviations seem to have a larger influence than maximum and minimum statistics. However, in some feature groups like the MST and the nearest neighbour group, the maximum statistics also provide considerable variations of feature values over α. These values provide suggestions for the best features to estimate the problem difficulty for Christofides. Furthermore, these features are similar to the most prominent features for 2-Opt [6] to a considerable extent. Nevertheless, there exist differences in the strength of the relationships and the contrast patterns of some features

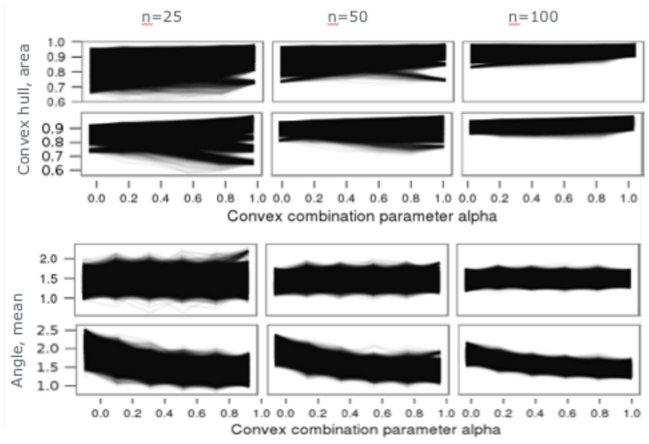


Figure 4: Some contrast patterns observed for Christofides (top) and 2-Opt (bottom)

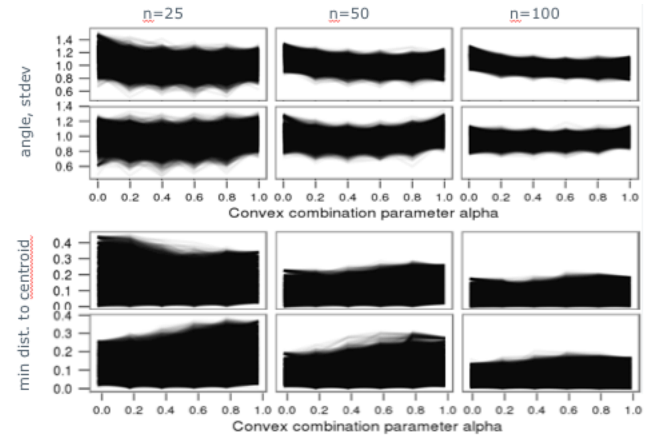


Figure 5: Some contrast patterns observed for 2-Approximation (top) and Christofides (bottom)

for the two algorithms such as the angle and the convex hull features (see Figure 4). As shown in Figure 5 such contrast patterns are also observed among the 2-Approximation and Christofides. This provides evidence on instance difficulty unique to the algorithm. Hence, we will further investigate this topic in next section, with the aim to identify complementary capabilities of the considered algorithms in this study.

4. PERFORMANCE COMPARISON OF LOCAL SEARCH AND APPROXIMATION ALGORITHMS

In this section, we consider algorithms with different underlying techniques, and we compare their relative performance on each others' difficult instances. We compare the Christofides algorithm, the 2-Opt local search algorithm studied in [6], and a well-known 2-Approximation algorithm. The 2-Approximation algorithm is similar to the Christofides algorithm but does not compute the minimum weight matching. Instead of this, it doubles the edges of the minimum spanning tree and uses the resulting graph to compute the Euler tour that is afterwards turned into a Hamiltonian cycle

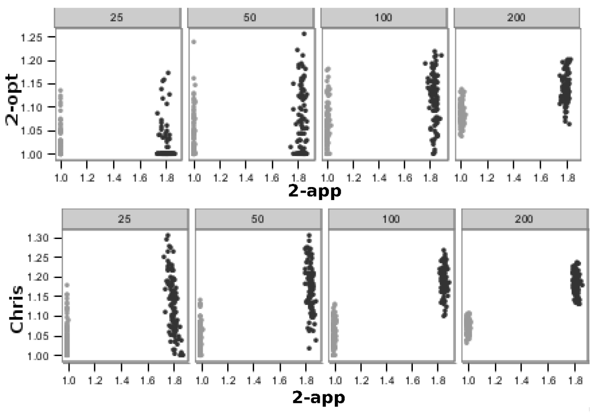


Figure 6: Performance of the 2-Opt algorithm (top) and Christofides algorithm (bottom) on the easy (grey) and hard (black) instances of the 2-Approximation algorithm.

by using short-cuts. A detailed description of this algorithm can be found in the textbook of Vazirani [12]. For the first test, these algorithms are considered pairwise, then run on each others' hard and easy instances, and the achieved approximation ratios are calculated. In this manner, it is possible to derive relative strengths and weaknesses of the considered algorithms by observing how well one algorithm performs in situations that are difficult for the others. Note that the three algorithms achieved approximation ratios close to 1 on their respective easy instances. On its hard instances, 2-Opt achieves ratios in the range from 1.15 to 1.3. The 2-Approximation algorithm achieves ratios of around 1.8 on its own hard instances, and the Christofides algorithm achieves ratios of roughly 1.4 on it own hard instances.

Our observations on the pair-wise comparison are as follows. As shown in Figure 6, both 2-Opt and Christofides algorithms perform better than the 2-Approximation algorithm on the hard instances generated by the 2-Approximation algorithm itself and worse on the easy ones. In the former case, for smaller instance sizes, 2-Opt achieves approximation ratios ranging from 1 to 1.2, and it slightly outperforms Christofides that achieves a range of 1 to 1.3. In contrast, for larger instance sizes, both algorithms have a similar performance with approximation ratios in a tighter range (1.1 to 1.2). As a general observation, 2-Opt and the Christofides algorithm perform similarly on the easy instances, despite the slight variations shown in the approximation values. The 2-Approximation algorithm obtains a better approximation ratio around 1, while the others range from 1 to 1.2 (see Figure 6).

As shown in Figure 7, both the 2-Approximation algorithm and the Christofides algorithm shows similar patterns of the achieved approximation ratios. For smaller instances, these values form straight lines for easy instances, and are scattered for hard instances. For the largest instances, two separate clusters are formed of the easy and hard instances. In the case of the hard 2-Opt instances, the ranges differ significantly for the two algorithms. There, the 2-Approximation algorithm obtains ratios ranging from 1 to 1.6, while the Christofides algorithm achieves a much smaller range from 1 to 1.2. Interestingly, the instances that are hard for 2-opt are not that hard for the Christofides algorithm (see bottom diagram in Figure 7): the approximation ratios obtained by 2-Opt on the hard instances vary from 1 to 1.3, whereas the range is

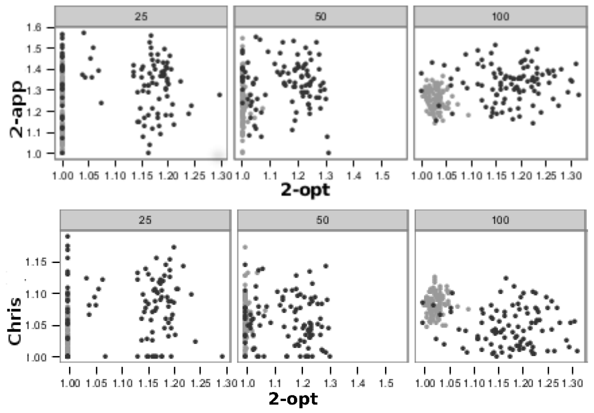


Figure 7: Performance of the 2-Approximation algorithm (top) and the Christofides algorithm (bottom) on the easy (grey) and hard (black) instances of 2-Opt algorithm.

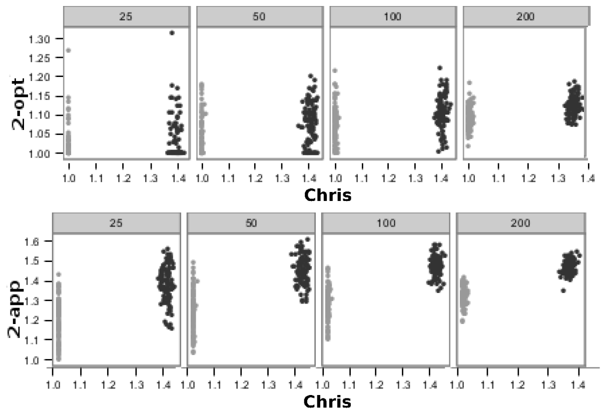


Figure 8: Performance of 2-Opt algorithm (top) and the 2-Approximation algorithm (bottom) on easy (grey) and hard (black) instances of Christofides algorithm.

1 to 1.15 for Christofides, and even just 1 to 1.1, for larger instances.

Figure 8 shows the results obtained by 2-Opt and the 2-Approximation algorithm on easy and hard Christofides instances. It can be observed that the Christofides algorithm itself has better performance for larger instance sizes than the 2-Approximation. This is more prominent in the case of hard instances where the clusters move to the upper right, with increasing instance size. Comparatively, 2-Opt achieves the best approximation values ranging from 1 to 1.3 for hard Christofides instances, the 2-Approximation algorithm (from 1.1 to 1.6) and Christofides (from 1.3 to 1.4). In contrast, Christofides achieves the best approximation ratios on its own easy instances (very close range around 1), whereas 2-Opt (from 1 to 1.2) and the 2-Approximation algorithm (from 1 to 1.5) cover wider ranges. For 2-Opt and the 2-Approximation algorithm these ranges get smaller with increasing instance size.

In summary it is observed that Christofides and 2-Opt algorithms surpass each other in the case of their own hard instances (see Figure 7 and 8), and this effect is more prominent in larger instance sizes. This implies that even though the hard instances of the different algorithms share some features, some other features are specific to each algorithm. For the easy instances, the generating algorithm generally performed best on its easy instances, and all algorithms achieve approximation ratios very close to 1. Considering hard instances only, the best approximation ratio is obtained by Christofides on the largest hard instances of 2-Opt (from 1 to 1.1). On the other hand, the worst approximation values for the hard instances are obtained for instances of the 2-Approximation algorithm by this algorithm itself, with values around 1.8. Both 2-Opt and Christofides algorithms compete with each other, having similar performances, while the 2-Approximation algorithm stays at a fair distance behind them. In general, these results imply that there are complementary capabilities of all the three algorithms on the difficult instances of each other. In addition, it becomes obvious that in general – as expected – hard and easy instances for a specific algorithm cannot be distinguished solely by means of the corresponding approximation ratios of the two remaining algorithms. The only slight exception is the 2-Approximation heuristic which allows for some conclusions regarding the approximation quality of the Christofides algorithm, especially for larger instance sizes.

5. CONCLUSIONS AND OUTLOOK

We used an evolutionary algorithm approach to generate easy and hard instances for the well-known Christofides and a 2-Approximation algorithm. Various features of easy and hard instances for the Christofides instances have been analysed in order to identify features for distinguishing the instance classes. Furthermore, the relationship of the feature values with the problem difficulty when moving from easy to hard instances has been examined which increased the understanding of underlying structures and relationships. Afterwards, we compared the Christofides, the 2-Approximation and a local search algorithm based on the 2-Opt operator by running the algorithms on each others' hard and easy instances. The results of this comparison of the hard instances point out complementary capabilities of the considered algorithms. Future work will be concentrated on feature based prediction of algorithm performance or the best suited algorithm for the analysed problem instances which will provide meaningful insights regarding algorithm design and especially the final goal of automated algorithm selection for given TSP instances.

References

[1] D. Applegate, W. J. Cook, S. Dash, and A. Rohe. Solution of a min-max vehicle routing problem. *INFORMS Journal on Computing*, 14(2):132–143, 2002.

[2] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

[3] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12. ACM, 2012.

[4] T. Kötzing, F. Neumann, H. Röglin, and C. Witt. Theoretical analysis of two aco approaches for the traveling salesman problem. *Swarm Intelligence*, pages 1–21, 2012.

[5] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21(1):498–516, 1973.

[6] O. Mersmann, B. Bischl, J. Bossek, H. Trautmann, M. Wagner, and F. Neumann. Local search and the traveling salesman problem: A feature-based characterization of problem hardness. In *Proceedings of the Learning and Intelligent Optimization Conference (LION)*, LNCS. Springer, 2012. http://arxiv.org/abs/1208.2318.

[7] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer, 2010.

[8] K. Smith-Miles and L. Lopes. Measuring instance difficulty for combinatorial optimization problems. *Computers & OR*, 39(5):875–889, 2012.

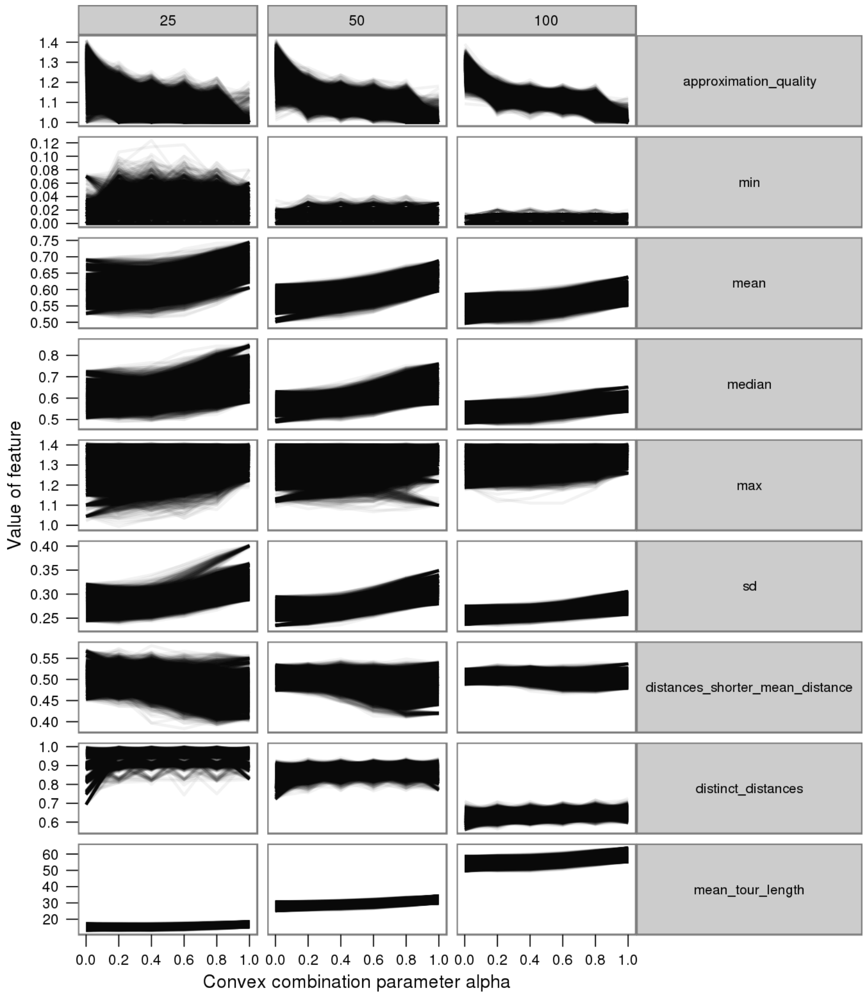
[9] K. Smith-Miles, J. I. van Hemert, and X. Y. Lim. Understanding tsp difficulty by learning from evolved instances. In *LION*, pages 266–280, 2010.

[10] A. Sutton and F. Neumann. A parameterized runtime analysis of evolutionary algorithms for the euclidean traveling salesperson problem. In *Proceedings of Association of Advancements of Artificial Intelligence*. AAAI, 2012.

[11] J. I. van Hemert. Evolving combinatorial problem instances that are difficult to solve. *Evol. Comput.*, 14(4):433–462, Dec. 2006.

[12] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.

[13] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *J. Artif. Int. Res.*, 32(1):565–606, June 2008.



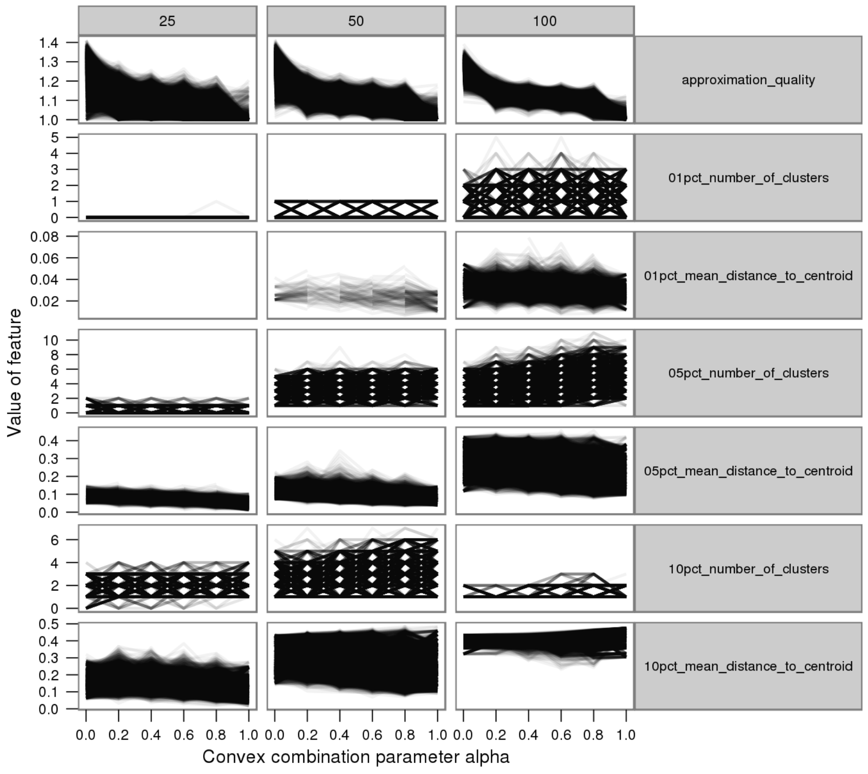
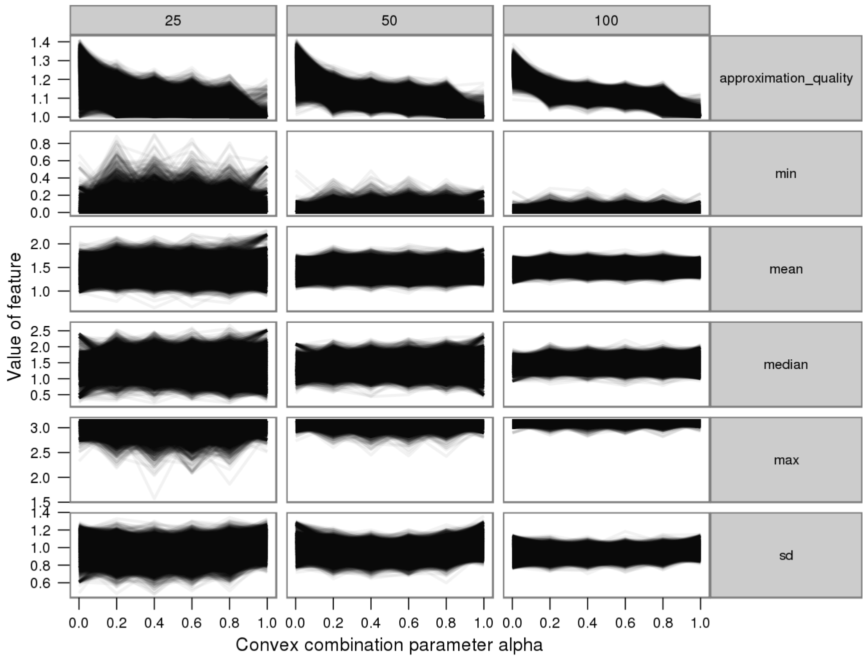


Figure 9: Distance features (top) and Cluster features (bottom): approximation quality and feature values for different α levels of all conducted morphing experiments for Christofides.



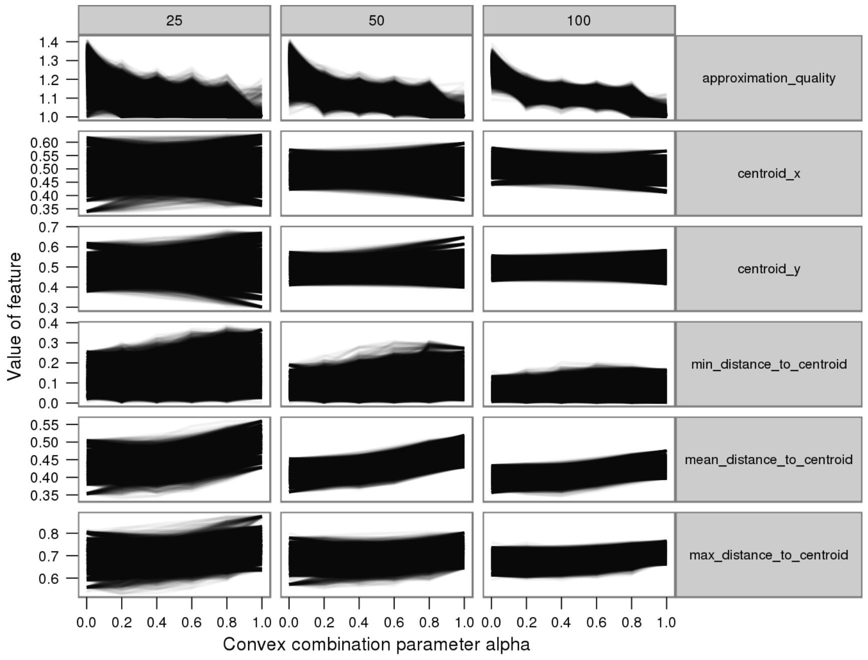
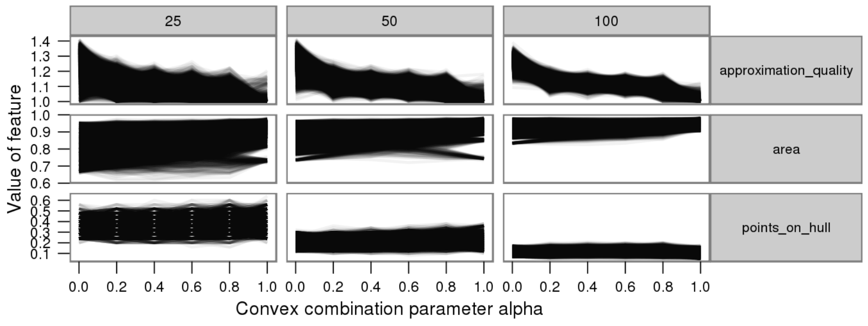


Figure 10: Angle (top) and Centroid Features (bottom): approximation quality and feature values for different α levels of all conducted morphing experiments for Christofides.



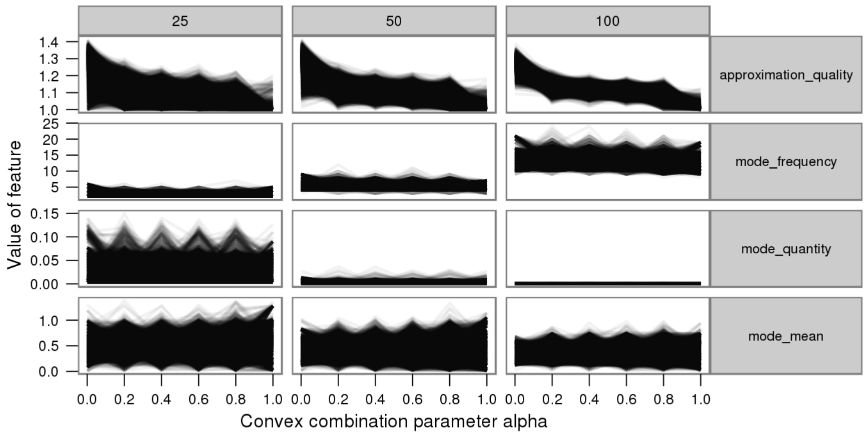


Figure 11: Convex Hull (top) and Mode (bottom) features: approximation quality and feature values for different α levels of all conducted morphing experiments for Christofides.

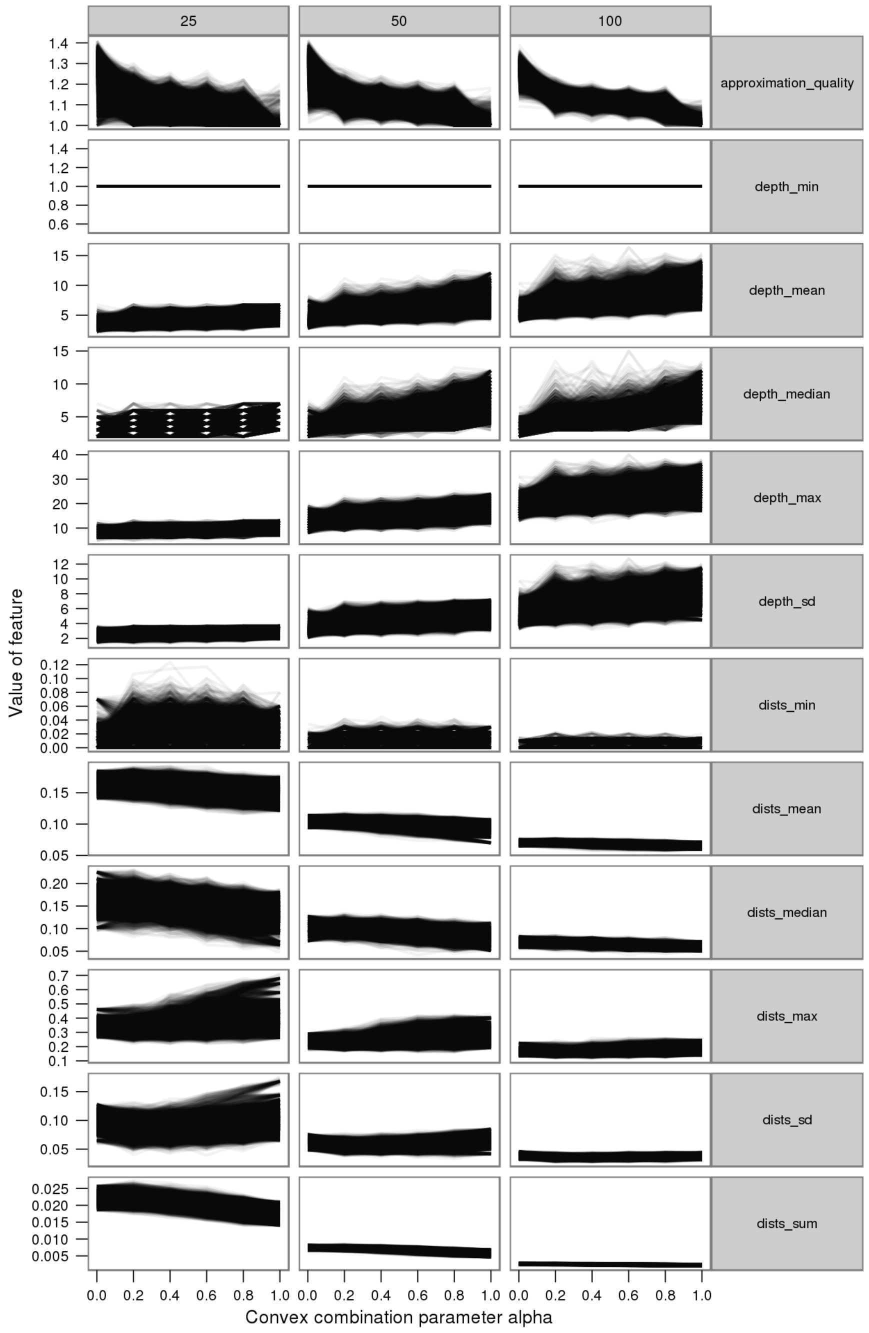


Figure 12: MST features: approximation quality and feature values for different α levels of all conducted morphing experiments for Christofides.

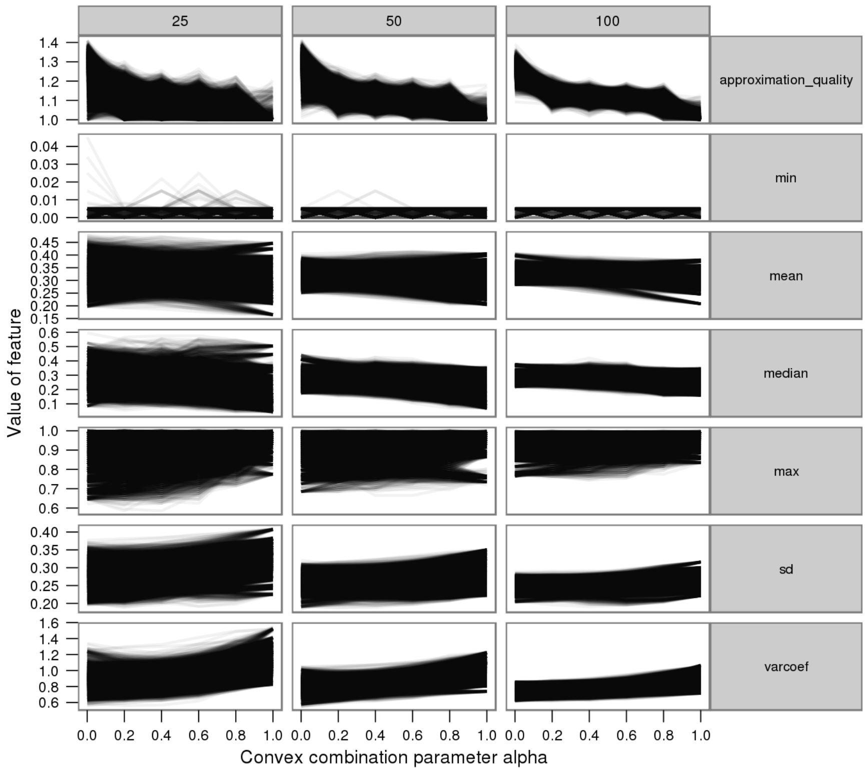


Figure 13: Nearest neighbour distance features: approximation quality and feature values for different α levels of all conducted morphing experiments for Christofides.