Feature-Based Diversity Optimization for Problem Instance Classification

Wanru Gao[1], Samadhi Nallaperuma[2], and Frank Neumann[1(✉)]

[1] School of Computer Science, The University of Adelaide, Adelaide, Australia
frank@cs.adelaide.edu.au

[2] Department of Computer Science, The University of Sheffield, Sheffield, UK

Abstract. Understanding the behaviour of heuristic search methods is a challenge. This even holds for simple local search methods such as 2-OPT for the Traveling Salesperson problem. In this paper, we present a general framework that is able to construct a diverse set of instances that are hard or easy for a given search heuristic. Such a diverse set is obtained by using an evolutionary algorithm for constructing hard or easy instances that are diverse with respect to different features of the underlying problem. Examining the constructed instance sets, we show that many combinations of two or three features give a good classification of the TSP instances in terms of whether they are hard to be solved by 2-OPT.

1 Introduction

Heuristic search methods such as local search, simulated annealing, evolutionary algorithms and ant colony optimization have been shown to be very successful for various combinatorial optimization problems. Although they usually don't come with performance guarantees on their runtime and/or approximation behaviour, they often perform very well in several situations. Understanding the conditions under which optimization algorithms perform well is essential for automatic algorithm selection, configuration and effective algorithm design. In both the artificial intelligence (AI) [1–4] and operational research communities [5,6], this topic has become a major point of interest.

The feature-based analysis of heuristic search algorithms has become an important part in understanding such type of algorithms [7,8]. This approach characterizes algorithms and their performance for a given problem based on features of problem instances. Thereby, it provides an important tool for bridging the gap between pure experimental investigations and mathematical methods for analysing the performance of search algorithms [9–11]. Current methods for the feature-based analysis are based on constructing hard and easy instances for an investigated search heuristic and a given optimization problem by evolving instances using an evolutionary algorithm [7,12,13]. This evolutionary algorithm constructs problem instances where the examined algorithm either shows a bad (good) approximation behaviour and/or requires a large (small) computational

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 869–879, 2016.
DOI: 10.1007/978-3-319-45823-6_81

effort to come up with good or optimal solutions. Although the evolutionary algorithm for constructing such instances is usually run several times to obtain a large set of hard (easy) instances, the question arises whether the results in terms of features of the instances obtained give a good characterization of problem difficulty.

In the paper, we propose a new approach of constructing hard and easy instances. Following some recent work on using evolutionary algorithms for generating diverse sets of instances that are all of high quality [14,15], we introduce an evolutionary algorithm which maximizes diversity of the obtained instances in terms of a given set of features. Our approach allows to generate a set of instances that is guaranteed to be diverse with respect to the problem features at hand. Carrying out this process for several combinations of features of the considered problem and algorithm gives a much better classification of instances according to their difficulty of being solved by the considered algorithm.

To show the benefit of our approach compared to previous methods, we consider the classical 2-OPT algorithm for the TSP. Previous feature-based analyses have already considered hard and easy instances in terms of approximation ratio and analyzed the features of such hard (easy) instances obtained by an evolutionary algorithm. The experimental results of our new approach show that diversity optimization of the features results in an improved coverage of the feature space over classical instance generation methods. In particular, the results show that for some combinations of two features it is possible to classify hard and easy instances into two clusters with a wider coverage of the feature space compared to the classical methods. Moreover, the three-feature combinations further improve the classification of hard and easy instances for most of the feature combinations. Furthermore, a classification model is built using these diverse instances that can classify TSP instances based on hardness for 2-OPT.

The remainder of this paper is organized as follows. Firstly, we introduce the Euclidean TSP and the background on feature based analysis. Afterwards, we state our diversity optimization approach for evolving instances according to feature values and report on the impact of diversity optimization in terms of the range of feature values. As feature values can be very diverse both for easy and hard instances, we consider the combinations of several features for instance classification afterwards. We then build a classification model that can classify instances based on hardness and finally finish with some conclusions.

2 Background

We consider the classical NP-hard Euclidean Traveling Salesperson problem (TSP) as the example problem for evolving hard and easy instances which have a diverse set of features. Our methodology can be applied to any optimization problem, but using the TSP in our study has the advantage that it has already been investigated extensively from different perspectives including the area of feature-based analysis.

The input of the problem is given by a set $V = \{v_1, \ldots, v_n\}$ of n cities in the Euclidean plane and Euclidean distances $d : V \times V \to \mathbb{R}_{\geq 0}$ between

Algorithm 1. $(\mu + \lambda)$-EA_D

1 Initialize the population P with μ TSP instances of approximation ratio at least α_h.
2 Let $C \subseteq P$ where $|C| = \lambda$.
3 For each $I \in C$, produce an offspring I' of I by mutation. If $\alpha_A(I') \geqslant \alpha_h$, add I' to P.
4 While $|P| > \mu$, remove an individual $I = \arg\min_{J \in P} d(J, P)$ uniformly at random.
5 Repeat step 2 to 4 until termination criterion is reached.

the cities. The goal is to find a Hamiltonian cycle whose sum of distances is minimal. A candidate solution for the TSP is often represented by a permutation $\pi = (\pi_1, \ldots, \pi_n)$ of the n cities and the goal is to find a permutation π^* which minimizes the tour length given by $c(\pi) = d(\pi_n, \pi_1) + \sum_{i=1}^{n-1} d(\pi_i, \pi_{i+1})$.

For our investigations cities are always in the normalized plane $[0, 1]^2$, i. e. each city has an x- and y-coordinate in the interval $[0, 1]$. In following, a TSP instance always consists of a set of n points in $[0, 1]^2$ and the Euclidean distances between them.

Local search heuristics have been shown to be very successful when dealing with the TSP and the most prominent local search operator is the 2-OPT operator [16]. The resulting local search algorithm starts with a random permutation of the cities and repeatedly checks whether removing two edges and reconnecting the two resulting paths by two other edges leads to a shorter tour. If no improvement can be found by carrying out any 2-OPT operation, the tour is called locally optimal and the algorithm terminates.

The key factor in the area of feature-based analysis is to identify the problem features and their contribution to the problem hardness for a particular algorithm and problem combination. This can be achieved through investigating hard and easy instances of the problem. Using an evolutionary algorithm, it is possible to evolve sets of hard and easy instances by maximizing or minimizing the fitness (tour length in the case of the TSP) of each instance [5–8]. However, none of these approaches have considered the diversity of the instances explicitly. Within this study we expect to improve the evolutionary algorithm based instance generation approach by introducing diversity optimization.

The structural features are dependent on the underlying problem. In [7], there are 47 features in 8 groups used to provide an understanding of algorithm performance for the TSP. The different feature classes established are distance features, mode features, cluster features, centroid features, MST features, angle features and convex hull features. The feature values are regarded as indicators which allow to predict the performance of a given algorithm on a given instance.

3 Feature-Based Diversity Optimization

In this section, we introduce our approach of evolving a diverse set of easy or hard instances which are diverse with respect to important problem features.

As in previous studies, we measure hardness of a given instance by the ratio of the solution quality obtained by the considered algorithm and the value of an optimal solution.

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 value of the solution produced by algorithm A for the given instance I, and $OPT(I)$ is value of an optimal solution for instance I. Within this study, $A(I)$ is the tour length obtained by 2-OPT for a given TSP instance I and $OPT(I)$ is the optimal tour length which we obtain in our experiments by using the exact TSP solver Concorde [17].

We propose to use an evolutionary algorithm to construct sets of instances of the TSP that are quantified as either easy or hard in terms of approximation and are diverse with respect to underlying features of the produced problem instances. Our evolutionary algorithm (shown in Algorithm 1) evolves instances which are diverse with respect to given features and meet given approximation ratio thresholds.

The algorithm is initialized with a population P consisting of μ TSP instances which have an approximation ratio at least α_h in the case of generating a diverse set of hard instances. In the case of easy instances, we start with a population where all instances have an approximation ratio of at most α_e and only instances of approximation ratio at most α_e can be accepted for the next iteration. In each iteration, $\lambda \leq \mu$ offspring are produced by selecting λ parents and applying mutation to the selected individuals. Offsprings that don't meet the approximation threshold are rejected immediately.

The new parent population is formed by reducing the set consisting of parents and offsprings satisfying the approximation threshold until a set of μ solutions is achieved. This is done by removing instances one by one based on their contribution to the diversity according to the considered feature.

The core of our algorithm is the selection among individuals meeting the threshold values for the approximation quality according to feature values. Let $I_1, \ldots, I_k$ be the elements of P and $f(I_i)$ be their features values. Furthermore, assume that $f(I_i) \in [0, R]$, i.e. feature values are non-negative and bounded above by R.

We assume that $f(I_1) \leq f(I_2) \leq \ldots \leq f(I_k)$ holds. The diversity contribution of an instance I to a population of instances P is defined as

$$d(I, P) = c(I, P),$$

where $c(I, P)$ is a contribution based on other individuals in the population

Let I_i be an individual for which $f(I_i) \neq f(I_1)$ and $f(I_i) \neq f(I_k)$. We set

$$c(I_i, P) = (f(I_i) - f(I_{i-1})) \cdot (f(I_{i+1}) - f(I_i)),$$

which assigns the diversity contribution of an individual based on the next smaller and next larger feature values. If $f(I_i) = f(I_1)$ or $f(I_i) = f(I_k)$, we

set $c(I_i, P) = R^2$ if there is no other individual $I \neq I_i$ in P with $f(I) = f(I_i)$ and $c(I_i, P) = 0$ otherwise. This implies an individual I_i with feature value equal to any other instances in the population gains $c(I_i, P) = 0$. Furthermore, an individual with the unique smallest and largest feature value always stays in the population when working with $\mu \geq 2$.

In [7], 47 features of TSP instances for characterizing easy and hard TSP instances have been studied. We consider 7 features coming from different feature classes which have shown to be well suited for classification and prediction. These features are: *angle_mean*, *centroid_mean_distance_to_centroid*, *chull_area*, *cluster_10pct_mean_distance_to_centroid*, *mst_depth_mean*, *nnds_mean* and *mst_dists_mean*.

We refer the reader to [7] for a detailed explanation for each feature. We carry out our diversity optimization approach for these features and use the evolutionary algorithm to evolve for each feature a diverse population of instances that meets the approximation criteria for hard/easy instances given by the approximation ratio thresholds.

All programs in our experiments are written in R and run in R environment [18]. We use the functions in tspmeta package to compute the feature values [7].

The setting of the evolutionary algorithm for diversity optimization used in our experiments is as follows. We use $\mu = 30$ and $\lambda = 5$ for the parent and offspring population size, respectively. The 2-OPT algorithm is executed on each instance I five times with different initial solutions and we set $A(I)$ to the average tour length obtained. The examined instance sizes n are 25, 50 and 100, which are denoted by the number of cities in one instance. Based on previous investigations in [7] and initial experimental investigations, we set $\alpha_e = 1$ for instances of size 25 and 50, and $\alpha_e = 1.03$ for instances of size 100. Evolving hard instances, we use $\alpha_h = 1.15, 1.18, 1.2$ for instances of size $n = 25, 50, 100$, respectively. The mutation operator picks in each step one city for the given parent uniformly at random and changes its x- and y-coordinator by choosing an offset according to the Normal-distribution with standard deviation σ. Coordinates that are out of the interval are reset to the value of the parent. Based on initial experiments we use two mutation operators with different values of σ. We use $\sigma = 0.025$ with probability 0.9 and $\sigma = 0.05$ with probability 0.1 in a mutation step. The evolutionary algorithm terminates after $10,000$ generations which allows to obtain a good diversity for the considered features. For each $n = 25, 50, 100$ and each of the 7 features, a set of easy and hard instances are generated, which results in 42 independent runs of the $(\mu+\lambda)$-EA$_D$.

4 Range of Feature Values

We first evaluate our diversity optimization approach in terms of the diversity that is obtained with respect to a single feature. Focusing on a single feature in each run provides the insight of the possible range of a certain feature value for hard or easy instances. The previous study [7], suggests that there are some

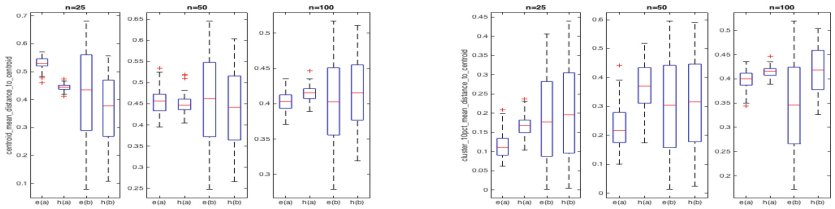

Fig. 1. (left) The boxplots for centroid mean distance to centroid feature values of a population consisting of 100 different hard or easy TSP instances of different number of cities without or with diversity mechnism. (right) The boxplots for cluster 10 % distance distance to centroid feature values of a population consisting of 100 different hard or easy TSP instances of different number of cities without or with diversity mechnism. Easy and hard instances from conventional approach and diversity optimization are indicated by e(a), h(a) and e(b), h(b) respectively.

differences in the possible range of feature values for easy and hard instances. We study the effect of the diversity optimization on the range of features by comparing the instances generated by diversity optimization to the instances generated by the conventional approach in [7]. Evolving hard instances based on the conventional evolutionary algorithm, the obtained instances have mean approximation ratios of 1.12 for $n = 25$, 1.16 for $n = 50$, and 1.18 for $n = 100$. For easy instances, the mean approximation ratios are 1 for $n = 25, 50$ and 1.03 for $n = 100$.

Figure 1 (left) presents the variation of the mean distance of the distances between points and the centroid feature (*centroid_mean_distance_to_centroid*) for hard and easy instances of the three considered sizes 25, 50 and 100. Each set consists of 100 instances generated by independent runs [7]. As shown in Fig. 1 (left) the hard instances have higher feature values than for easy instances for all instance sizes. For example, for instance size 100 and for the hard instances the median value (indicated by the red line) is 0.4157 while its only 0.0.4032 for the easy instances. The respective range of the feature value is 0.0577 for the hard instances and 0.0645 for the easy instances. For the instances generated by diversity optimization (easy and hard instances are indicated by e(b) and h(b) respectively), there is a difference in the median feature values for the hard and easy instances similar to the instances generated by the conventional approach. Additionally, the range of the feature values for both the hard and easy instances has significantly increased. For example, for the instance size 100, the median value for easy instances is 0.4028 and the range is 0.2382. For the hard instances of the same size, the median is 0.04157 while the range is 0.1917 (see Fig. 1 (left)).

Similarly, Fig. 1 (right) presents the variation of cluster 10 % distance to centroid (*cluster_10pct_distance_to_centroid*) feature for the hard and easy instances generated by the conventional approach (indicated by (e(a) and h(a)) and for the hard and easy instances generated by diversity optimization (indicated by

(e(b) and h(b))). The general observations from these box plots are quite similar to the observations from the *mst_dist_mean* shown in Fig. 1 (left).

The above results suggest that the diversity optimization approach has resulted in a significant increase in the coverage over the feature space. Having the threshold for approximation ratios (α_e and α_h) our method guarantees the hardness of the instances. These approximation thresholds are more extreme than the mean approximation values obtained by the conventional method. Being able to discover all these instances spread in the whole feature space, our approach provides a strong basis for more effective feature based prediction.

As a result of the increased ranges and the similar gap in median feature values for hard and easy instances compared to the conventional instances, there is a strong overlap in the ranges of the features for easy and hard instances generated by the diversity optimization. This is observed in the results for *mst_dist_mean* and *cluster_10pct_distance_to_centroid* shown in Fig. 1. Similar pattern holds for the other features as well. This prevents a good classification of problem instances based on single feature value.

5 Classification Based on Multiple Features

As a single feature is not capable in clearly classifying the hard/easy instances, combinations of two or three different features are examined in the following. Our analysis mainly focuses on combinations of the 7 previously introduced features.

According to the observation and discussion in [7], the two features *distance_max* and *angle_mean* can be considered together to provide an accurate classification of the hard and easy instances. Whereas after increasing the diversity over the seven different feature values and a wider coverage of the 2D space is achieved, the separation of easy and hard instances is not so obvious, as shown in Fig. 2. There are large overlapping areas lying between the two groups of instances. As the number of cities in an instance increases, the overlapping area becomes larger. It is hard to do classification based on this. Therefore the idea of combining three different feature is put forward.

Support vector machines (SVMs) are well-known supervised learning models in machine learning which can be used for classification, regression and outliers

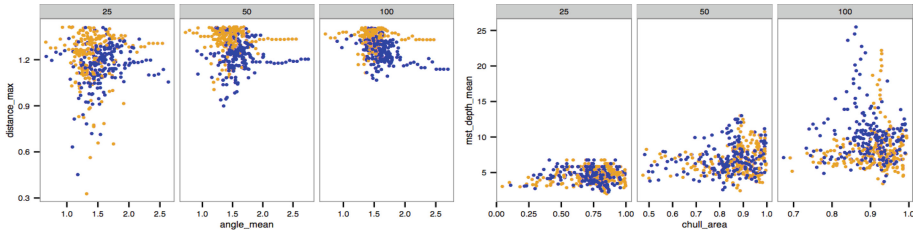

Fig. 2. 2D Plots of feature combinations which provide a separation between easy and hard instances. The blue dots and orange dots represent hard and easy instances respectively. (Color figure online)

detection [19]. In order to quantify the separation between instances of different hardness based on the feature values, SVM models are constructed for each combination of features.

Let ACC_n be the training accuracy of a feature combination in separating the hard and easy instances of size n. We define ACC_n as the ratio of number of instances which are correctly classified by the model to the total number of instances in the dataset. All classification experiments are done in R with library$\{e1071\}$ [20].

The training data of the SVM models are the population of 420 instances generated as in Sect. 3 and the training accuracy is regarded as a quantified measurement of the separation between hard and easy instances. The feature combinations used for classification are the 21 two-feature combinations and 35 three-feature combinations discussed in Sect. 5.

The linear classifier is the first model tried in classifying the dataset. Since the dataset is not linearly separable, taken the trade-off between maximizing the margin and minimizing the number of misclassified data points into consideration, the soft-margin SVM is used for classification. From experiment results, most of the accuracies of different feature combinations lie in the range of 0.6 to 0.7, which implies the high possibility that the linear models are not suitable for separating the hard and easy instances based on most of the feature combinations. Therefore we move to applying kernel functions for non-linear mapping of the feature combination. The Radial Basis Function (RBF) kernel is one of the well-known kernel functions used in SVM classification.

There are two parameters needed when applying RBF kernel, which are C(cost) and γ. The parameter setting for RBF is crucial, since increasing C and γ leads to accurate separation of the training data but at the same time causes

Table 1. The accuracy of SVM with RBF kernel separating the hard and easy instances in different two-feature space.

Feature 1	Feature 2	ACC_{25}	ACC_{50}	ACC_{100}
angle_mean	centroid_mean_distance_to_centroid	0.8476	0.8071	0.8071
angle_mean	chull_area	0.7857	0.7810	0.7929
angle_mean	cluster_10pct_mean_distance_to_centroid	0.7810	0.7786	0.8000
angle_mean	mst_depth_mean	0.7524	0.7381	0.8000
angle_mean	nnds_mean	0.8167	0.8833	0.8452
angle_mean	mst_dists_mean	0.8119	0.8024	0.8405
centroid_mean_distance_to_centroid	chull_area	0.8619	0.7667	0.8381
centroid_mean_distance_to_centroid	cluster_10pct_mean_distance_to_centroid	0.8524	0.8357	0.7548
centroid_mean_distance_to_centroid	mst_depth_mean	0.8381	0.7643	0.8095
centroid_mean_distance_to_centroid	nnds_mean	0.8786	0.9524	0.8476
centroid_mean_distance_to_centroid	mst_dists_mean	0.8905	0.8571	0.8762
chull_area	cluster_10pct_mean_distance_to_centroid	0.8000	0.7881	0.8548
chull_area	mst_depth_mean	0.7429	0.7429	0.7571
chull_area	nnds_mean	0.8071	0.8905	0.8452
chull_area	mst_dists_mean	0.8619	0.8643	0.9024
cluster_10pct_mean_distance_to_centroid	mst_depth_mean	0.7619	0.7714	0.7929
cluster_10pct_mean_distance_to_centroid	nnds_mean	0.8190	0.8833	0.8643
cluster_10pct_mean_distance_to_centroid	mst_dists_mean	0.8095	0.8095	0.8738
mst_depth_mean	nnds_mean	0.7786	0.8595	0.8405
mst_depth_mean	mst_dists_mean	0.8095	0.8214	0.8810
nnds_mean	mst_dists_mean	0.8500	0.9143	0.9024

Table 2. The accuracy of SVM with RBF kernel separating the hard and easy instances in different three-feature space.

Feature 1	Feature 2	Feature 3	ACC_{25}	ACC_{50}	ACC_{100}
angle_mean	centroid_mean_distance_to_centroid	chull_area	0.9500	0.9190	0.9452
angle_mean	centroid_mean_distance_to_centroid	cluster_10pct_mean_distance_to_centroid	0.9405	0.9357	0.8214
angle_mean	centroid_mean_distance_to_centroid	mst_depth_mean	0.9548	0.9548	0.9214
angle_mean	centroid_mean_distance_to_centroid	nnds_mean	0.9452	0.9952	0.9833
angle_mean	centroid_mean_distance_to_centroid	mst_dists_mean	0.9571	0.9500	0.9524
angle_mean	chull_area	cluster_10pct_mean_distance_to_centroid	0.9524	0.9310	0.8881
angle_mean	chull_area	mst_depth_mean	0.9357	0.9238	0.9500
angle_mean	chull_area	nnds_mean	0.9405	0.9714	0.9571
angle_mean	chull_area	mst_dists_mean	0.9667	0.9619	0.9143
angle_mean	cluster_10pct_mean_distance_to_centroid	mst_depth_mean	0.9214	0.9143	0.9810
angle_mean	cluster_10pct_mean_distance_to_centroid	nnds_mean	0.9476	0.9690	0.9333
angle_mean	cluster_10pct_mean_distance_to_centroid	mst_dists_mean	0.9571	0.9143	0.9405
angle_mean	mst_depth_mean	nnds_mean	0.9310	0.9762	0.9238
angle_mean	mst_depth_mean	mst_dists_mean	0.9476	0.9262	0.9476
angle_mean	nnds_mean	mst_dists_mean	0.9429	0.9762	0.8833
centroid_mean_distance_to_centroid	chull_area	cluster_10pct_mean_distance_to_centroid	0.9476	0.9333	0.9310
centroid_mean_distance_to_centroid	chull_area	mst_depth_mean	0.9595	0.8762	0.9762
centroid_mean_distance_to_centroid	chull_area	nnds_mean	0.9667	0.9881	0.9929
centroid_mean_distance_to_centroid	chull_area	mst_dists_mean	0.9714	0.9714	0.8381
centroid_mean_distance_to_centroid	cluster_10pct_mean_distance_to_centroid	mst_depth_mean	0.9476	0.9286	0.8571
centroid_mean_distance_to_centroid	cluster_10pct_mean_distance_to_centroid	nnds_mean	0.9643	0.9905	0.8810
centroid_mean_distance_to_centroid	cluster_10pct_mean_distance_to_centroid	mst_dists_mean	0.9500	0.9595	0.9190
centroid_mean_distance_to_centroid	mst_depth_mean	nnds_mean	0.9500	0.9881	0.9595
centroid_mean_distance_to_centroid	mst_depth_mean	mst_dists_mean	0.9548	0.9548	0.9595
centroid_mean_distance_to_centroid	nnds_mean	mst_dists_mean	0.9667	1.0000	0.9952
chull_area	cluster_10pct_mean_distance_to_centroid	mst_depth_mean	0.9286	0.9524	0.9333
chull_area	cluster_10pct_mean_distance_to_centroid	nnds_mean	0.9524	0.9667	0.9667
chull_area	cluster_10pct_mean_distance_to_centroid	mst_dists_mean	0.9595	0.9595	0.9929
chull_area	mst_depth_mean	nnds_mean	0.9381	0.9857	0.9476
chull_area	mst_depth_mean	mst_dists_mean	0.9476	0.9738	0.9833
chull_area	nnds_mean	mst_dists_mean	0.9714	0.9857	0.9667
cluster_10pct_mean_distance_to_centroid	mst_depth_mean	nnds_mean	0.9214	0.9857	0.9738
cluster_10pct_mean_distance_to_centroid	mst_depth_mean	mst_dists_mean	0.9500	0.9476	0.9643
cluster_10pct_mean_distance_to_centroid	nnds_mean	mst_dists_mean	0.9643	0.9833	0.9976
mst_depth_mean	nnds_mean	mst_dists_mean	0.9429	0.9929	0.9929

over-fitting. The SVMs here are generated for quantifying the separation rate between hard and easy instances rather than classifying other instances. After some initial trials, (C, γ) is set to $(100, 2)$ in all the tests to avoid over-fitting. This parameter setting may not be the best for the certain feature combination in SVM classifying, but it helps us to gain some understanding of the separation of hard and easy instances generated from previous experiments based on the same condition.

Tables 1 and 2 show the accuracy of different two-feature or three-feature combination in hard and easy instances separation. With RBF kernel, SVM with certain parameter setting can generate a model separating the dataset with average accuracy of 0.8170, 0.8244 and 0.8346 in 2D feature space for instance size 25, 50 and 100 respectively. Whereas with three features, SVM with the same parameter setting provides a separation with average accuracy of 0.9503, 0.9584 and 0.9422 for instance size 25, 50 and 100 respectively.

From the results, it can be concluded that there are better separations between hard and easy instances in the 3D feature space.

6 Conclusions

With this paper, we have introduced a new methodology of evolving easy/hard instances which are diverse with respect to feature sets of the optimization problem at hand. Using our diversity optimization approach we have shown that the easy and hard instances obtained by our approach covers a much wider range in the feature space than previous methods. The diversity optimization approach provides instances which are diverse with respect to the investigated features. The proposed population diversity measurements provide good evaluation of the diverse over single or multiple feature values. Our experimental investigations for 2-OPT and TSP have shown that our large set of diverse instances can be classified quite well into easy and hard instances when considering a suitable combination of multiple features which provide some guidance for predication as the next step. In particular, the SVM classification model built with the diverse instances that can classify TSP instances based on problem hardness provides a strong basis for future performance prediction models that lead to automatic algorithm selection and configuration. Building such models would require further experimentation to determine the minimal set of strong features that can predict performance accurately.

Acknowledgement. This research has been supported by the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement no. 618091 (SAGE) and by the Australian Research Council under grant agreement DP140103400.

References

1. Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K.: Algorithm runtime prediction: methods & evaluation. Artif. Intell. **206**, 79–111 (2014)

2. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artif. Intell. Rev. **18**(2), 77–95 (2002)
3. Eggensperger, K., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Efficient benchmarking of hyperparameter optimizers via surrogates. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 25–30 January 2015, Austin, Texas, USA, pp. 1114–1120 (2015)
4. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 25–30 January 2015, Austin, Texas, USA, pp. 1128–1135 (2015)
5. Smith-Miles, K., Lopes, L.: Measuring instance difficulty for combinatorial optimization problems. Comput. Oper. Res. **39**(5), 875–889 (2012)
6. van Hemert, J.I.: Evolving combinatorial problem instances that are difficult to solve. Evol. Comput. **14**(4), 433–462 (2006)
7. Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J., Neumann, F.: A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. Ann. Math. Artif. Intell. **69**(2), 151–182 (2013)
8. Smith-Miles, K., van Hemert, J., Lim, X.Y.: Understanding TSP difficulty by learning from evolved instances. In: Blum, C., Battiti, R. (eds.) LION 4. LNCS, vol. 6073, pp. 266–280. Springer, Heidelberg (2010)
9. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity, 1st edn. Springer, New York (2010)
10. Kötzing, T., Neumann, F., Röglin, H., Witt, C.: Theoretical analysis of two ACO approaches for the traveling salesman problem. Swarm Intell. **6**(1), 1–21 (2012)
11. Englert, M., Röglin, H., Vöcking, B.: Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP. Algorithmica **68**(1), 190–264 (2014)
12. Nallaperuma, S., Wagner, M., Neumann, F., Bischl, B., Mersmann, O., Trautmann, H.: A feature-based comparison of local search and the Christofides algorithm for the travelling salesperson problem. In: FOGA 2013, pp. 147–160 (2013)
13. Nallaperuma, S., Wagner, M., Neumann, F.: Parameter prediction based on features of evolved instances for ant colony optimization and the traveling salesperson problem. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) PPSN 2014. LNCS, vol. 8672, pp. 100–109. Springer, Heidelberg (2014)
14. Ulrich, T., Bader, J., Thiele, L.: Defining and optimizing indicator-based diversity measures in multiobjective search. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 707–717. Springer, Heidelberg (2010)
15. Ulrich, T., Bader, J., Zitzler, E.: Integrating decision space diversity into hypervolume-based multiobjective search. In: GECCO, pp. 455–462 (2010)
16. Croes, G.A.: A method for solving traveling-salesman problems. Oper. Res. **6**(6), 791–812 (1958)
17. Applegate, D., Cook, W., Dash, S., Rohe, A.: Solution of a min-max vehicle routing problem. INFORMS J. Comput. **14**(2), 132–143 (2002)
18. Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2015)
19. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
20. Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F.: e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.6-7 (2015)