Automated Algorithm Selection in Single-Objective Continuous Optimization: A Comparative Study of Deep Learning and Landscape Analysis Methods

Raphael Patrick Prager[1(✉)], Moritz Vinzent Seiler[1], Heike Trautmann[1,2], and Pascal Kerschke[3]

[1] Data Science: Statistics and Optimization, University of Münster, Münster, Germany
{raphael.prager,moritz.seiler,heike.trautmann}@uni-muenster.de
[2] Data Management and Biometrics, University of Twente, Enschede, The Netherlands
[3] Big Data Analytics in Transportation, TU Dresden, Dresden, Germany
pascal.kerschke@tu-dresden.de

Abstract. In recent years, feature-based automated algorithm selection using exploratory landscape analysis has demonstrated its great potential in single-objective continuous black-box optimization. However, feature computation is problem-specific and can be costly in terms of computational resources. This paper investigates feature-free approaches that rely on state-of-the-art deep learning techniques operating on either images or point clouds. We show that point-cloud-based strategies, in particular, are highly competitive and also substantially reduce the size of the required solver portfolio. Moreover, we highlight the effect and importance of cost-sensitive learning in automated algorithm selection models.

Keywords: Automated algorithm selection · Exploratory landscape analysis · Deep learning · Continuous optimization

1 Introduction

The algorithm selection problem (ASP), nowadays, is a well-studied topic [12,31]. Essentially, it boils down to the identification of a mechanism $m : \mathcal{I} \rightarrow \mathcal{A}$, which selects an optimal algorithm a out of a collection of algorithms $\mathcal{A}$ for any given optimization problem $i \in \mathcal{I}$. Typically, this mechanism m is in need of a numerical representation of a problem instance i to make an informed decision. In the domain of single-objective continuous optimization, this role has largely been filled by exploratory landscape analysis (ELA) [21].

Various research endeavours have improved and evaluated ELA for automated algorithm selection (AAS) [3,14,29]. In this paper, we offer an alternative

G. Rudolph et al. (Eds.): PPSN 2022, LNCS 13398, pp. 3–17, 2022.
https://doi.org/10.1007/978-3-031-14714-2_1

and conceptually very different means to characterize a problem instance i in the domain of AAS. We call this collection of methods 'feature-free'. While these methods still require a sample of the search space, they do not require any computation of numerical features. Instead, the initial sample of a search space is used to construct a $2D$ image (fitness map) or a cloud of points (fitness cloud), which assists in the algorithm selection process. In a recent publication, we evaluated a few of these methods for AAS limited to $2D$ single-objective continuous problems with promising results [28]. In a subsequent publication, we successfully tested these methods for predicting a landscape's general characteristics, e.g., the degree of multi-modality, without any dimensionality restrictions [33].

This paper combines the gained insights from both works by investigating the methods' potential for AAS while simultaneously bypassing the limiting factor that $2D$ fitness maps are inherently restricted to $2D$ problems. To compare our results with existing research, we mimic the algorithm selection scenario of Kerschke and Trautmann [14]. Thereby, we will demonstrate the potential of our feature-free and deep learning (DL) based approaches for AAS and promote further research within this domain.

This paper is structured as follows. First, we briefly introduce ELA as well as the construction of the fitness maps and fitness clouds in Sect. 2. Thereafter, we describe our underlying data sets and the experimental setup in Sect. 3. This is followed by a discussion of the results in Sect. 4, as well as a summary of our findings and an outlook on further research potential in Sect. 5.

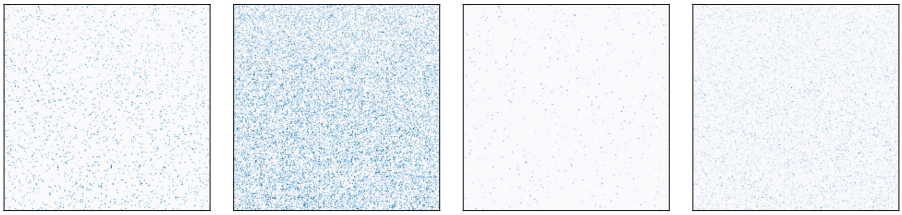
2 Background

2.1 Exploratory Landscape Analysis

ELA features [15,20] numerically characterize the landscape of continuous, single-objective optimization problems which is crucial for problem understanding and as input to automated algorithm selection models [12,23,24]. ELA features are required to be descriptive, fast to compute, and reasonably cheap, i.e., the size of the initial sample on which they are most commonly based, has to be small [4,13]. Over the years, a large variety of feature sets emerged, of which we specifically consider the following:

Classical ELA comprises 22 features of six categories. While the meta model, level set and y-distribution features are commonly used, the remaining ones require additional function evaluations [21]. Six **Fitness Distance Correlation** features provide metrics describing the distances in decision and objective space, and the relation of these distances in-between those two spaces [11]. The 16 **Dispersion** features divide the initial sample into different subsets w.r.t. sample quantiles and contrast homogeneity of both groups [18]. Five **Information Content** features rely on smoothness, ruggedness and neutrality measures of random walks over the initial sample [22]. **Nearest Better Clustering** (5 features) derives several metrics and ratios on nearest (better) neighbour distances of the initial sample [13]. Moreover, the **Miscellaneous** feature group summarizes 10 features of different concepts, e.g. principal component analysis based features or the problem dimensionality [15].

2.2 Fitness Map



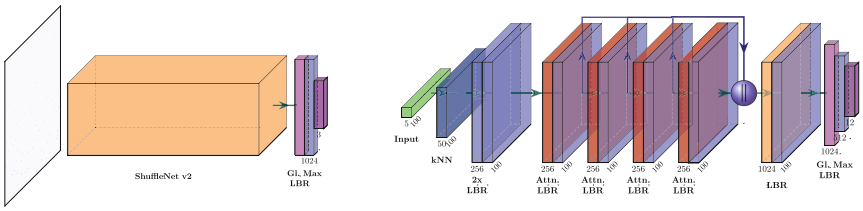
a) $5D$: reduced MC b) $10D$: reduced MC c) $5D$: Conv-rMC d) $10D$: Conv-rMC

Fig. 1. Examples of different fitness maps. In particular, visualization of the rMC (a–b) and Conv-rMC (c–d) image-based approaches for different dimensionalities. Note, that we colored and inverted the color-scale for illustrative purposes.

Several works [2,28,32,33] have recently proposed measures to alleviate problems with feature-based AAS as it comes with three major drawbacks: instance features (1) are manually designed in an elaborative process, (2) require additional calculations which may increase selection costs, and (3) are limited to a specific domain and, thus, cannot be adapted to other problems easily (see [32]). These three drawbacks are alleviated by applying DL-based approaches directly on the raw instance information and, thereby, avoiding the need for computing instance features as well as circumnavigating the three drawbacks.

In our previous work, we have proposed several feature-free approaches [28, 33]. These approaches range from simple convolutional neural networks (CNN) [16] to point cloud transformers (PCT) [6]. The former approach is based on images while the latter one is based on point clouds.

In [33], we proposed four different techniques to create fitness maps based on an initial sample $\mathcal{X}$ of the search space with $\mathcal{X} \in \mathbb{R}^{m \times (D+1)}$. Here, m denotes the sample size and $D+1$ represents the size of the decision space in addition to the fitness value. We formally define a fitness map as $\mathcal{F}_{map} \in \mathbb{R}^{l \times l \times 1}$ with a width and height of $l \times l$ pixels plus a singular color channel. Each technique consists of two steps. First, the D-dimensional samples and their fitness values are normalized and, then, the samples are projected into $2D$ by using one of four proposed dimensionality reduction techniques (cf. [33] for details about the normalization and choice of parameters like l). Afterwards, the fitness values are mapped into the respective $2D$-Cartesian plane at their corresponding transformed location. This $2D$-plane can then be interpreted as a gray scaled image in which unknown or bad fitness values have a brighter hue and respectively good values a darker one (see Fig. 1). These four proposed dimensionality reduction techniques are: (1) classical principal component analysis (PCA) [26], (2) PCA including the fitness-value (PCA-Func), (3) multi channel (MC), and (4) reduced multi channel (rMC). The MC approaches create an individual $2D$ feature map for each possible pairwise combination of search space variables which amounts



a) Image-based **b)** Point cloud-based

Fig. 2. Visualizations of the image- (left) and point-based approach (right), adapted from [33]. a) A ShuffleNet V2 [19], extended by the following layers: Global Max. Pooling, Linear, Batch Normalization [10], ReLU [25](LBR), and Linear with Softmax activation. b) kNN-Embedding, followed by four Attention Layers (Attn.), a Global Max. Pooling, LBR, and Linear Layer with Softmax activation as proposed in [33].

to $n = \binom{D}{2}$ total feature maps. These are consolidated into a single fitness map of shape $\mathcal{F}_{map} \in \mathbb{R}^{l \times l \times n}$ with n color channels. The rMC approach projects these n feature maps into a single one by mean aggregation for each pixel of the n fitness maps instead of adding additional color channels. Thereby, rMC fitness maps retain their original shape which make them invariant to the problem dimensionality while also decreasing the number of channels and weights. For more details on the methods, we refer to [33]. Extending our previous work, we introduce an additional image-based approach below.

Convolutional Reduced Multi Channel (Conv-rMC). This approach works in a similar manner as the rMC approach. First, an individual $2D$-plane is created for every pairwise combination of decision variables, resulting in n planes. As explained in [33], the major drawback of the previously considered MC approach is the exponential growth of the number $n = \binom{D}{2}$ of $2D$-planes (and thus channels) w.r.t. the number of dimensions D. This has also been adressed by rMC, which reduces the amount of images n to a single $2D$-plane. Yet, the aggregation may cause the individual $2D$-planes to become indistinguishable from one-another [33]. Now, the herein proposed Conv-rMC approach is supposed to solve that issue. Conv-rMC projects the n images into a single, gray-scale image using a 1×1-convolutional layer, followed by a batch normalization layer [10] and a ReLU activation [25]. We choose this setup because this is identical to the main building blocks of the ShuffleNet v2 [19] architecture which we used for our experimental study. The additional weights of the convolutional layer and the batch normalization layer are trained by backpropagation. Representative fitness maps of the rMC and Conv-rMC can be found in Fig. 1.

2.3 Fitness Cloud

There are two significant issues related to image-based DL: (1) the resolution of the $2D$-planes is limited by the number of pixels as well as the upper and lower

bound of the optimization problem's decision space; and (2) for $D > 2$, some (not all) image-based approaches lose information due to our employed methods to reduce dimensionality [33].

Therefore, in [33], we explored the potential of a novel DL approach, called point cloud analysis. The advantage is that for a point cloud, DL can be directly applied to the individual observations of a sample without the need to project them into a $2D$ Cartesian plane first. Further, as most DL approaches for point clouds can handle any finite number of dimensions, there is no information loss for $D > 2$ (cf. Fig. 2). We formally define a point cloud as $\mathcal{F}_{cloud} \in \mathbb{R}^{m \times (D+1)}$ where m is the size of our initial sample and $D + 1$ are the dimensions plus the fitness value. Next, as the resolution is not limited by the number of the image's pixels, local neighborhoods can be processed accurately in every detail. Yet, DL for point clouds must respect the point-order isomorphic of the input data which increases the complexity of DNNs, substantially. For details, we refer to [33].

Although there are several different point cloud approaches available, in [33] we chose point cloud transformers (PCT) because of their transformer background. Transformers are part of DL and were introduced by [36] for neural language processing. They are in-particular great in capturing global relationships which is important for tasks like *high-level property prediction* or *automated algorithm selection*. In addition, [33] proposed a novel embedding technique which is based on the nodes in a k-nearest neighbor (kNN) graph. This embedding technique is supposed to embed every observation of $\mathcal{X}$ into the context of its local neighborhood. The embedding layer is represented as the first layer in Fig. 2 b) in dark blue. This is necessary as transformers are good in capturing global relations but may lose attention to local neighborhood [34]. The embedding layer comes with three additional hyperparameters: (1) k for the kNN-search, (2) p for the L_p-Norm which is used as distance measure, and (3) Δ_{max} to limit the local neighborhood. To avoid poorly chosen parameter values, we performed hyperparameter optimization for the three parameters during our experimental study (see Sect. 3.3).

3 Experiments

3.1 Algorithm Performance Features

Commonly used in the continuous optimization community to evaluate their algorithms is the Black-Box Optimization Benchmark (BBOB) [9]. This benchmark suite is embedded into the platform Comparing Continuous Optimizers (COCO) [8], which provides data from various competitions on the BBOB test bed. The accompanying results of these competitions are uploaded to COCO and offer further opportunity for scrutiny and study.

To test and compare our feature-free methods w.r.t. automated algorithm selection, we use data of Kerschke and Trautmann [14], which in turn was collected from COCO. This data set consists of the performance data of twelve different optimization algorithms evaluated on BBOB. An enumeration of these algorithms can be found in Fig. 4 or in [14]. BBOB is structured as a set of

different problem **functions** (FID) which can be initialized in different **dimensions** (D), and can also be subjected to slight modification such as shifts and rotations. The latter leads to different **instances** (IID) of a given problem function. In total, the set of benchmark problem consists of 480 unique problem instances i, constituted by the tuple $i := (\text{FID}, \text{D}, \text{IID})$ with $\text{FID} \in \{1, 2, ..., 24\}$, $D \in \{2, 3, 5, 10\}$, and $\text{IID} \in \{1, 2, ..., 5\}$.

For a given problem instance i, an algorithm is considered to be successful if it reaches the objective value of the global optimum up to a given precision value. In [14], the authors deemed a precision value of 10^{-2} as reasonable. A smaller precision value would otherwise lead to many unsuccessful runs across all algorithms. For a given FID and D, the five different runs (over the instances) are aggregated using the expected running time (ERT) [7].

$$ERT = \frac{1}{s} \sum_{k} FE_k, \tag{1}$$

where k refers to the different BBOB instances, FE_k denotes the spent function evaluations on k, and s is the number of successful runs. Since we aggregated over the instances, the algorithmic performance data set is reduced from 480 to $480/5 = 96$ observations.

While an ERT is useful in its own right, it is hard to compare these absolute ERT values between functions of different dimensionality simply because algorithms are allowed a larger budget in higher dimensions. As a remedy, we employ a slight adaption which is called relative ERT (relERT). For any given problem $p := (\text{FID}, \text{D})$, we divide the ERT values of the twelve algorithms by its respective lowest one. Thereby, we obtain a measure which indicates by which factor each algorithm is more expensive (in terms of function evaluations) compared to the best performing algorithm on that specific problem p. Furthermore, there are cases where none of the five instances of a problem was solved by an algorithm. This causes the value s (from Eq. 1) to be zero, resulting in an infinite ERT. For the purpose of our algorithm selector, we impute these values similar to Kerschke and Trautmann [14] by taking the largest observed (i.e., finite) relERT value overall and multiplying it by the factor 10 as penalty [14].

The single-best solver (SBS) across all problems is the HCMA with an relERT value of 30.37 [17]. This offers ample opportunity to close the gap between the SBS and theoretically achievable performance of 1. The latter is also referred to as the virtual-best solver (VBS). The VBS serves as an upper bound whereas the SBS is a baseline which needs to be surpassed.

3.2 Landscape Features

Here, the term 'landscape features' comprises each kind of input to our models, i.e., either a set of ELA features, a fitness map, or a fitness cloud. While these three input variants differ substantially, the underlying sample, upon which they are computed, largely remains the same. Meaning, these methods act as a surrogate of an individual problem instance i, which is constructed from an initial

sample of size $50D$. The fitness cloud differs in this respect because the sample size is a constant of either 100 or 500 – the lower ($D = 2$) and upper bound ($D = 10$) of our sampling sizes – to estimate the lower and upper performance bounds. This is done for every problem instance i, i.e., we create 480 different samples.

We use latin hypercube sampling which is inherently stochastic and aim to avoid propagating this stochasticity to the training phase of our algorithm selectors as much as is reasonably possible by sampling independently ten times for each i. Thereby, we not only reduce the stochasticity but also artificially increase our training data by the factor ten. This is especially beneficial since DL generally requires a more extensive set of training data. In short, we increase the number of observations for a specific surrogate variant from 480 to 4 800 ($= 480 \cdot 10$) observations. This is in stark contrast to the work of Kerschke and Trautmann [14]. There, the numerical landscape features where computed once for each problem instance i and aggregated using the median to match the performance data set with 96 observations. Here, we increase the performance data to match our landscape feature data set. To give an example, let us assume that for a given problem p (e.g., FID 1 in dimension 2 of BBOB), we have an algorithm a which performs best in terms of relERT. We surmise that this fact also holds true for the underlying five BBOB instances and we duplicate these relERT values onto these instances. With that, we increase the algorithm performance data set from 96 observations to $480 (= 96 \cdot 5)$ and duplicate these entries ten times, to match our landscape data set of size 4 800.

We compute the numerical landscape features, i.e., features described in Sect. 2.1, with the Python package `pflacco`[1]. Features which suffer from missing values are removed in their entirety. These belong mostly to the set of *ELA level* and sum up to 14 features. After removal, we are left with 48 distinct features. On the other hand, we create the fitness map as described in Sect. 2.2 with no further adjustment and an image resolution of 224×224 pixels. The fitness clouds, however, require the selection of a predetermined and static input size. For this, we ascertain two variants 100 and 500 as adequate based on [33].

3.3 Experimental Setup

We model our algorithm selection problem as a classification task with a typical data set of $\mathcal{S} = (X, y)$. Meaning, our algorithm selection models, which are described in more detail in the following, do not predict the performance for a given algorithm and problem instance, but rather predict which algorithm to choose for a given problem instance. Hence, our final data set does not consist of twelve algorithms for each problem instance, but only the best one on that specific problem. In this scenario, the algorithm operates as a label y, whereas the landscape features represent X within our machine learning model. In cases of misclassification, however, it is apparent that not each false prediction is equally costly. Therefore, we utilize a cost matrix C, where an individual value $c_{i,a}$

[1] https://github.com/Reiyan/pflacco_experiment.

defines the cost of a chosen algorithm a on a problem instance i. This granularity of costs, which is relegated to individual observations, is also known as example-specific cost-sensitivity [35] (see Eq. 2).

The performance of each used model is evaluated by using five-fold cross validation. Since our data set is comprised of five instances per BBOB function, each fold consists of the predetermined instance of each function.

Models Based on ELA Features. We designed two classification models based on ELA features. The first approach is similar to our previous work in [14], i.e., we use a gradient boosting classifier, a support vector machine (SVM) and a random forest (RF) in conjunction with feature selection. These experiments are conducted in Python where the machine learning models stem from `sklearn` [27]. Furthermore, we use a 'greedy forward-backward' feature selection strategy which is implemented in the package `mlxtend` [30]. This process deems most of the features unnecessary or redundant, reducing the set to 15 features. As feature selection is a computationally expensive task, we did not perform any hyper-parameter tuning (similar to [14]). A major distinction compared to the remaining models is that the aforementioned three classifiers are not cost-sensitive during their training. Through this, we intend to highlight the potential of cost-sensitivity on an individual observation-level.

The alternative strategy uses a multilayer-perceptron (MLP) [5] incorporating cost-sensitive training. Feature selection is omitted, as the training duration of an MLP is significantly larger than that other machine learning models, e.g., RF, and the MLP internally includes feature selection due to its working mechanism. In general, the models predict a probability distribution $\hat{p}_a \sim \hat{P}$ where $\hat{p}_a$ is the probability to choose the a-th solver out of all $\mathcal{A}$ solvers. To implement example-specific cost-sensitive training, we choose the loss function

$$\mathcal{L}_{\mathcal{A}}(\hat{P}, T, C; \Theta) = \sum_{a=1}^{\mathcal{A}} |t_{i,a} - \hat{p}_{i,a}| \cdot c_{i,a}. \tag{2}$$

Here, $t_{i,a} \in T_i$ are the true labels for the best performing algorithm on instance i, $c_{i,a} \in C$ are the costs for predicting algorithm a based on the current instance i, and Θ are the model's weights. The loss function has two properties: (1) it is continuously differentiable, and (2) $\mathcal{L}_K = 0$ for models that predict the best performing solver with a confidence of 1.

Next, we performed hyperparameter optimization using a coarse grid search. We tested for the learning rate $\in \{0.01, 0.001, 0.0001\}$, dropout $\in \{0.1, 0.3, 0.5\}$, the number of layers $\in [1, 6]$, hidden neurons $\in \{128, 256, 512, 1024\}$, and the learning rate decay $\in \{0.99, 0.97, 0.95\}$. The remaining setup (including the 5-fold cross validation) was chosen identically to the setup of the classical machine learning approach. We found that a model with 1 024 hidden neurons, 3 repetitions, a dropout of 0.1, trained with a learning rate of 0.01, and a learning rate decay of 0.99 performed best. In the following, we will refer to this model as ELA-MLP. All DL models were trained on a single Nvidia Quadro RTX 6000,

and by using the `PyTorch` library in Python. In addition, the MLP models were trained for 500 epochs and the models with the lowest loss on the validation folds were selected as final.

Models Based on Fitness Maps. The training setup of the image-based DL models is similar as proposed in [33]. We used a *ShuffleNet V2* [19] (see Fig. 2 a) with a width-multiplicator of 1.5, and we used the same cost-sensitive loss function for these models (see Eq. 2). Further, we tested several different training approaches: learning rate $\in \{0.001, 0.0003, 0.0001, 0.00005, 0.00001\}$, an entropy regularization $\in \{0, 0.001, 0.01, 0.1, 1, 10\}$, and a scaled and un-scaled version of the loss function. The remaining training setup was identical to the ELA-MLP approach. We found, that a learning rate of 0.0003, no entropy regularization, and the un-scaled loss worked best. All image-based models were trained for 100 epochs, and the models with the lowest loss on the validation folds were selected as the final models.

Models Based on Fitness Clouds. The model topology and training setup for the PCTs was chosen identically to [33] (see Fig. 2 b). We tested for different kNNs, $k \in \{3, 5, 10\}$, and (as proposed by [1]) also considered $p = 0.5$ for the L_p distance function. However, we could not find any performance difference between the different PCTs, confirming the finding of [33] that the hyperparameters of the embedding layer have (at most) only a slight impact on performance for the BBOB data. The PCT models were trained for 50 epochs and the models with the lowest loss on the validation folds were chosen again as final models.

Next, due to a limitation of our implementation, we were not able to train the PCT models with $50D$ points but had to choose a consistent number of points for all dimensions. By using 100 and 500 points, we can estimate the upper and lower PCT model performance as $50D = 100$ for 2D and $50D = 500$ for 10D. As PCTs have lower or higher sampling costs compared to all other models, they may have an advantage or disadvantage.

4 Results and Discussion

Table 1 summarizes the results of the considered approaches for constructing AAS models. Model performances are provided in terms of mean relERT values, split by dimensionality and BBOB function groups as well as in total, i.e., considering all instances simultaneously. Next to the SBS (cf. 3.1), results of feature-based approaches are contrasted to feature-free concepts based on fitness maps and point clouds. Figure 3 visualizes the respective results on problem dimension level in terms of a parallel-coordinate-plot. Except from the SBS, the relERT values of the AAS models incorporate the costs of the initial sample.

It becomes obvious that the classical ELA feature based AAS model approach (ELA-RF) is largely outperformed. However, this is the only strategy not considering cost-sensitive learning in terms of integrating the loss in relERT induced

Table 1. Performance results of difference AAS models represented by their relERT. The first two columns represent information about the underlying problems, i.e., the different dimensions (D) and the BBOB function groups (FGrp) of similar characteristics. The best option based on the ELA features (ELA-RF and ELA-MLP) for a given row is highlighted in green, and the best model consisting of either fitness map (columns MC to PCA-Func) or fitness cloud (PCT-100 and PCT-500) is highlighted in blue. Values displayed in red indicate the overall best option for a given group of problems.

D	FGrp	SBS	ELA-RF	ELA-MLP	MC	Conv-rMC	rMC	PCA	PCA-Func	PCT-100	PCT-500
2	1	3.71	10.41	10.59	23.95	23.95	23.95	14.99	15.12	23.95	61.15
	2	5.80	8.51	3.72	3.54	3.54	3.54	6.97	6.40	3.54	11.04
	3	6.29	1473.16	4.72	4.02	4.02	4.02	4.98	5.64	4.02	16.11
	4	25.34	3.89	9.25	8.90	8.90	8.90	26.98	24.00	8.90	12.21
	5	44.95	148.39	3.32	4.33	4.33	4.33	30.01	30.97	4.33	6.18
	all	17.69	342.22	6.43	9.17	9.17	9.17	17.19	16.84	9.17	21.77
3	1	356.10	1480.68	11.87	16.58	16.56	191.76	365.54	344.65	13.32	39.24
	2	4.46	8.33	3.50	3.32	3.32	3.51	5.32	5.20	2.85	6.65
	3	4.98	7.07	3.82	3.61	3.58	4.11	5.21	5.51	2.72	9.58
	4	2.63	441.96	5.06	11.16	11.54	9.36	2.77	3.11	11.49	11.87
	5	66.81	1.22	2.54	2.53	2.53	2.53	50.46	50.20	2.46	3.04
	all	90.43	403.67	5.44	7.61	7.68	43.87	89.22	84.92	6.72	14.39
5	1	11.99	14.14	11.97	22.69	22.69	22.70	22.89	22.80	16.27	33.39
	2	3.90	369.26	2.62	4.74	4.78	4.60	4.64	4.70	4.25	5.66
	3	4.21	150.44	3.97	6.72	6.72	6.49	6.40	6.56	5.21	9.23
	4	4.29	1470.28	6.81	4.38	4.38	4.38	4.38	4.38	4.33	4.47
	5	7.67	1.13	1.83	4.22	7.80	3.38	4.08	4.90	7.72	7.93
	all	6.52	402.38	5.56	8.71	9.46	8.46	8.64	8.83	7.69	12.41
10	1	2.74	14.64	15.27	16.34	16.34	16.34	16.39	16.41	5.46	16.34
	2	2.16	1.62	1.76	2.71	2.71	2.71	2.69	2.67	2.27	2.71
	3	2.76	2.87	4.35	4.48	4.48	4.48	4.48	4.48	3.10	4.48
	4	2.02	442.01	1.96	2.09	2.09	2.09	2.07	2.09	2.03	2.09
	5	23.64	148.01	3.25	21.77	23.74	14.61	5.08	10.21	23.66	23.74
	all	6.85	126.84	5.46	9.76	10.17	8.27	6.29	7.36	7.51	10.17
all	1	93.63	379.97	12.43	19.89	19.88	63.68	104.96	99.74	14.75	37.53
	2	4.08	96.93	2.90	3.58	3.59	3.59	4.91	4.75	3.23	6.52
	3	4.56	408.38	4.21	4.71	4.70	4.78	5.27	5.55	3.76	9.85
	4	8.57	589.54	5.77	6.63	6.73	6.18	9.05	8.40	6.69	7.66
	5	35.77	74.69	2.74	8.22	9.60	6.21	22.41	24.07	9.54	10.22
	all	30.37	318.78	5.72	8.81	9.12	17.44	30.34	29.49	7.78	14.68

by wrong predictions of the best suited solver. We exemplary report on ELA-RF performance, as the best option among alternative standard classifiers such as SVM or gradient boosting, which qualitatively show comparable performance. However, integrating cost-sensitive learning into a multilayer-perceptron model based on ELA features (ELA-MLP) has a tremendously positive impact. ELA-

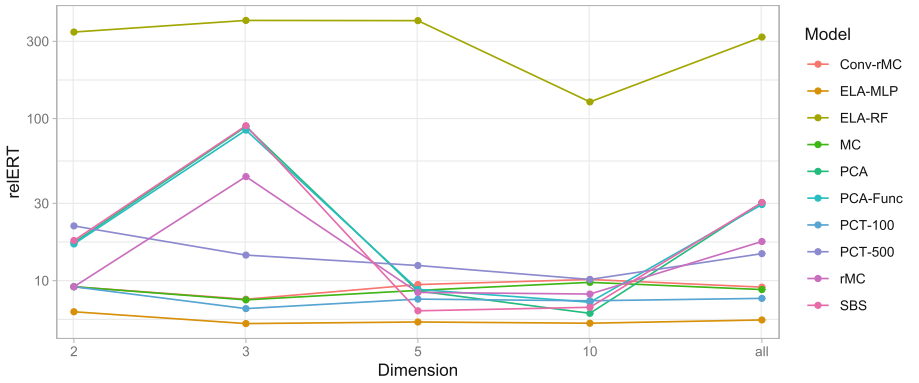


Fig. 3. Parallel coordinate plot visualizing AAS model results per problem dimension in terms of relERT based on Table 1.

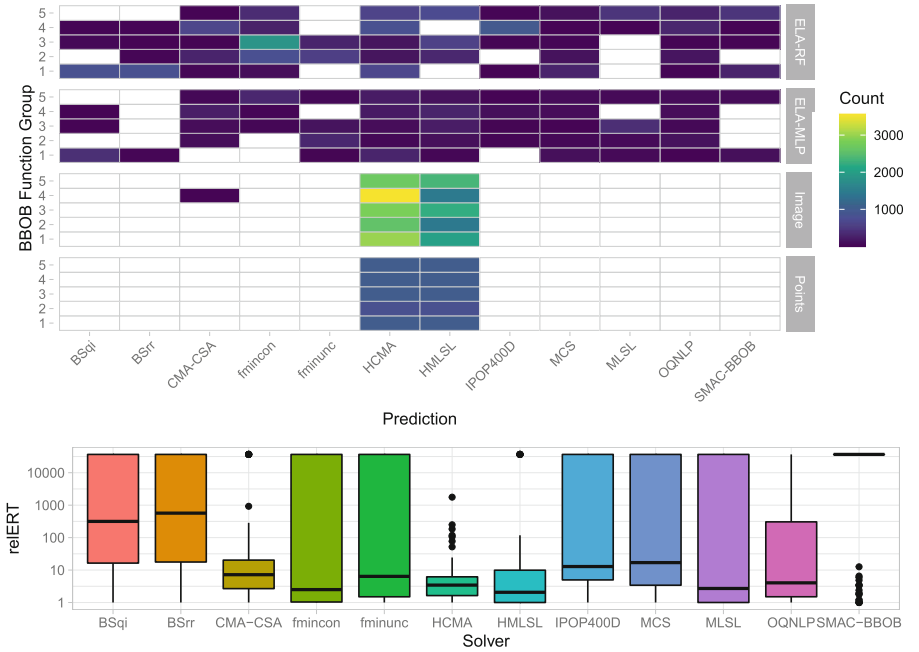


Fig. 4. Top Figure: Absolute frequencies of predicted solvers by AAS model type ('actual' solver portfolio) grouped by BBOB function groups. Bottom Figure: Individual solver performances (relERT) across all instances and dimensions.

MLP not only beats the SBS on almost all instances but rather largely reduces the SBS-VBS gap (across all instances and dimensionalities) by 78%. While this by itself is an outstanding result, even more notable is the performance of our feature-free alternatives. Without requiring feature information, the point-cloud

based PCT-100 model is second-best with really comparable results, closing the respective SBS-VBS gap by 74%. Surprisingly, within this model class, this PCT variant outperforms the more costly PCT-500 approach, i.e., the additional information stemming from the larger initial sample (for $D < 10$) does not outweigh the additional costs. Also, the fitness map based models MC- and Conv-rMC do not lag behind by much, while rMC and the PCA-based models do not show consistent quality across all dimensions.

Shifting the focus from pure performance analysis to a deeper understanding of model decisions, we see from Fig. 4, that models based on ELA features make use of the whole extent of the solver portfolio, whereas image- and point-cloud based strategies simultaneously manage to meaningfully reduce the size of the *required* solver portfolio. The upper image explicitly counts how often a specific solver has been selected within the AAS models, split by model type. Note that all image based approaches are aggregated within the *image* row, while the same holds for the *point* category. Interestingly, basically two solvers, i.e., HCMA and HMLSL, are sufficient to yield highly beneficial AAS decisions. A third, less important candidate is CMA-CSA. This is supported by the bottom figure displaying solver performances across all instances in terms of boxplots. The three respective solvers are performing consistently well, but also are most robust in terms of substantially smaller variability in terms of relERT values.

5 Conclusions and Outlook

We introduced a conceptually novel feature-free alternative to the conventional numerical landscape features and, thereby, provide an innovative solution to the ASP. Although the feature-free models perform slightly worse than the ELA feature models, they significantly outperform the single-best solver of our algorithm portfolio and meaningfully reduce the size of the algorithm portfolio.

However, our feature-free methods offer huge potential for future research. As discussed in [33], a considerable challenge of our proposed image-based methods is the balancing act between information loss for higher dimensions (which holds for PCA, PCA-Func, MC, and rMC) and increasing sparsity of points in lower dimensions (which holds for MC and Conv-rMC). Thus, for small dimensions most of the provided feature maps are empty and the weights get partially trained, only. Therefore, we proposed an alternative point cloud-based approach (PCT) which can also be considered as highly competitive to the conventional approaches. Summarizing, the classical ELA-MLP approach shows a slightly better overall AAS model performance compared to PCT-100, but comes with the trade-off of requiring a larger solver portfolio.

However, there is also large improvement potential for PCT based methods. First, the embedding layer may play a crucial role for the model's performance, yet we know little about the impact of the different hyperparameters. Thus, we will closer investigate the respective impact and experiment with different distance metrics. Also, we will investigate the potential and usability of deep features for cross-domain knowledge transfer. Deep features can be obtained after

the last hidden layer of a trained model applied to a new task. Afterwards, a small MLP or any other ML algorithm can be trained efficiently on the obtained deep features. This is especially useful if the quantity of data or the training resources for the new tasks are limited.

References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 420–434. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44503-X_27
2. Alissa, M., Sim, K., Hart, E.: Algorithm selection using deep learning without feature extraction. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 198–206 (2019)
3. Bischl, B., Mersmann, O., Trautmann, H., Preuß, M.: Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, pp. 313–320 (2012)
4. Bossek, J., Doerr, C., Kerschke, P.: Initial design strategies and their effects on sequential model-based optimization: an exploratory case study based on BBOB. In: Proceedings of the 22nd Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 778–786 (2020)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
6. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: PCT: point cloud transformer. Comput. Visual Media **7**(2), 187–199 (2021)
7. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-Parameter Black-Box Optimization Benchmarking 2010: Experimental Setup. Research Report RR-7215, INRIA (2010). https://hal.inria.fr/inria-00462481
8. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. Optimi. Meth. Software **36**(1), 114–144 (2021)
9. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Technical report RR-6829, INRIA (2009). https://hal.inria.fr/inria-00362633/document
10. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)
11. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the 6th International Conference on Genetic Algorithms (ICGA), pp. 184–192. Morgan Kaufmann Publishers Inc. (1995)
12. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: survey and perspectives. Evol. Comput. (ECJ) **27**(1), 3–45 (2019)
13. Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: Detecting funnel structures by means of exploratory landscape analysis. In: Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 265–272. ACM, July 2015

14. Kerschke, P., Trautmann, H.: Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. Evol. Comput. (ECJ) **27**(1), 99–127 (2019)
15. Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. In: Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., Vichi, M. (eds.) Applications in Statistical Computing. SCDAKO, pp. 93–123. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25147-5_7
16. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. Handbook Brain Theory Neural Networks **3361**(10), 1995 (1995)
17. Loshchilov, I., Schoenauer, M., Sèbag, M.: Bi-population CMA-ES algorithms with surrogate models and line searches. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation. GECCO 2013 Companion, pp. 1177–1184. ACM (2013)
18. Lunacek, M., Whitley, L.D.: The dispersion metric and the CMA evolution strategy. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 477–484. ACM (2006)
19. Ma, N., Zhang, X., Zheng, H.-T., Sun, J.: ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 122–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_8
20. Malan, K.M., Engelbrecht, A.P.: A survey of techniques for characterising fitness landscapes and some possible ways forward. Inf. Sci. (JIS) **241**, 148–163 (2013)
21. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 829–836. ACM (2011). Recipient of the 2021 ACM SigEVO Impact Award
22. Muñoz Acosta, M.A., Kirley, M., Halgamuge, S.K.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE Trans. Evol. Comput. (TEVC) **19**(1), 74–87 (2015)
23. Muñoz Acosta, M.A., Sun, Y., Kirley, M., Halgamuge, S.K.: Algorithm selection for black-box continuous optimization problems: a survey on methods and challenges. Inf. Sci. (JIS) **317**, 224–245 (2015)
24. Muñoz, M.A., Kirley, M.: Sampling effects on algorithm selection for continuous black-box optimization. Algorithms **14**(1), 19 (2021). https://doi.org/10.3390/a14010019
25. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: ICML 2010, Madison, WI, USA, pp. 807–814. Omnipress (2010)
26. Pearson, K.: On lines and planes of closest fit to system of points in space. Philos. Mug 6th ser. **2**, 559–572 (1901)
27. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
28. Prager, R.P., Seiler, M.V., Trautmann, H., Kerschke, P.: Towards feature-free automated algorithm selection for single-objective continuous black-box optimization. In: Proceedings of the IEEE Symposium Series on Computational Intelligence. Orlando, Florida, USA (2021)
29. Prager, R.P., Trautmann, H., Wang, H., Bäck, T.H.W., Kerschke, P.: Per-instance configuration of the modularized CMA-ES by means of classifier chains and exploratory landscape analysis. In: Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI), pp. 996–1003. IEEE (2020)

30. Raschka, S.: MLxtend: providing machine learning and data science utilities and extensions to python's scientific computing stack. J. Open Source Software (JOSS) **3**(24), 638 (2018)
31. Rice, J.R.: The algorithm selection problem. Adv. Comput. **15**(65–118), 5 (1976)
32. Seiler, M., Pohl, J., Bossek, J., Kerschke, P., Trautmann, H.: Deep learning as a competitive feature-free approach for automated algorithm selection on the traveling salesperson problem. In: Bäck, T., et al. (eds.) PPSN 2020. LNCS, vol. 12269, pp. 48–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58112-1_4
33. Seiler, M.V., Prager, R.P., Kerschke, P., Trautmann, H.: A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes. arXiv preprint (2022)
34. Shaw, P., Uszkoreit, J., Vaswani, A.: Self-Attention with Relative Position Representations. arXiv preprint arXiv:1803.02155 (2018)
35. Turney, P.D.: Types of Cost in Inductive Concept Learning. arXiv preprint cs/0212034 (2002)
36. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)