The Importance of Landscape Features for Performance Prediction of Modular CMA-ES Variants

Ana Kostovska, Diederick Vermetten, Sašo Džeroski, Carola Doerr, Peter Korosec, Tome Eftimov

The Importance of Landscape Features for Performance Prediction of Modular CMA-ES Variants

Ana Kostovska
Jožef Stefan Institute &
Jožef Stefan International
Postgraduate School
Ljubljana, Slovenia

Diederick Vermetten
Leiden Institute for Advanced
Computer Science
Leiden, The Netherlands

Sašo Džeroski
Jožef Stefan Institute &
Jožef Stefan International
Postgraduate School
Ljubljana, Slovenia

Carola Doerr
Sorbonne Université, CNRS, LIP
Paris, France

Peter Korosec
Jožef Stefan Institute
Ljubljana, Slovenia

Tome Eftimov
Jožef Stefan Institute
Ljubljana, Slovenia

ABSTRACT

Selecting the most suitable algorithm and determining its hyper-parameters for a given optimization problem is a challenging task. Accurately predicting how well a certain algorithm could solve the problem is hence desirable. Recent studies in single-objective numerical optimization show that supervised machine learning methods can predict algorithm performance using landscape features extracted from the problem instances.

Existing approaches typically treat the algorithms as black-boxes, without consideration of their characteristics. To investigate in this work if a selection of landscape features that depends on algorithms' properties could further improve regression accuracy, we regard the modular CMA-ES framework and estimate how much each landscape feature contributes to the best algorithm performance regression models. Exploratory data analysis performed on this data indicate that the set of most relevant features does not depend on the configuration of individual modules, but the influence that these features have on regression accuracy does. In addition, we have shown that by using classifiers that take the features' relevance on the model accuracy, we are able to predict the status of individual modules in the CMA-ES configurations.

KEYWORDS

evolutionary computation, exploratory landscape analysis, modular CMA-ES

1 INTRODUCTION

With the growth of the field of optimization, many algorithms have been built upon again and again, leading to vast families of algorithms that share a core design aspect. While often-times, these variants are introduced on their own, combining their contributions can allow for the creation of many more novel algorithm configurations. In particular, this can be achieved by modularizing these contributions and allowing these modules to be swapped out at will, essentially creating a configurable algorithm design space based upon a specific core algorithm, such as Genetic Algorithms [3, 38], Simulated Annealing [10], Particle Swarm Optimization (PSO) [7], and hybrids between Differential Evolution and PSO [6].

In this work, we focus on the family of Covariance Matrix Adaptation Evolution Strategies (CMA-ES) [15]. We make use of the ModCMA [8, 33], which implements 11 modules that can be combined arbitrarily to create thousands of variants of CMA-ES. The modules available range from elitism in the selection procedure, mirrored and orthogonal sampling mechanisms [1, 37] to local restart strategies with changing population sizes [2, 11]. Even though all of these methods have been proposed in isolation, it has been shown that combinations of these modules can lead to configurations of the modular CMA-ES which significantly outperform all of the base variants [34].

Several studies explored the family of modular CMA-ES in different learning settings including exploratory analysis of CMA-ES [9], automated algorithm performance prediction [31], automated algorithm selection [16], and automated algorithm configuration [5, 29].

De Nobel et al. [9] analyzed the CMA-ES behavior by using time-series features extracted from its dynamic strategy parameters. Their results showed that these features can be used to classify isolated CMA-ES modules and have the potential to be used in the prediction of their performance.

Trajnov et al. [31] learned an explainable automated performance prediction model by using the landscape features of the problem instances [24] to predict the quality of the solution after some fixed budget (i.e., number of function evaluations). Independent models were trained for different modular CMA-ES configurations and each configuration was further represented using the Shapley values that provide explanations on global (i.e., all problem instances that were involved) and local level (i.e., for each problem instance separately). Such representations allow them to distinguish between different modular CMA-ES configurations.

Jankovic and Doerr [16] used the modular CMA-ES to perform automated algorithm selection in a fixed-budget setting. By using the landscape features of the problem instances they learned performance regression models that were further used to select the appropriate modular CMA-ES configuration.

All aforementioned studies are only a part of a wide range of studies that contribute to understanding the behavior of modular CMA-ES [12, 32, 35, 36]. Even though great effort has been put in this research direction, most of the studies in automated algorithm performance prediction and selection are treating the CMA-ES configuration as a black-box system. That is, they do not aim at exploring the impact that each module has on the final performance of the CMA-ES configuration. In [9], the time-series features have been used to classify only the isolated CMA-ES modules, but there

is no information on how these features are linked when those modules are combined to create a new CMA-ES configuration. Prager et al. [29] investigated a problem instance-based configuration model that selects the optimal CMA-ES modules to create a configuration specific to a given problem instance. For this purpose, they trained random forest classifiers into a classifier chain scenario that uses the landscape features of the problem instance. The model is trained in a supervised learning setting and can predict if some modules should be activated or not. This study allows a selection of CMA-ES modules utilizing the landscape features of the problem instances, however without explanation of which landscape features are important for each module separately.

Our contribution: In this paper, we propose an approach for exploring which landscape features calculated for the problem instances are contributing to the performance prediction of the CMA-ES modules. Compared with the work done in [29], where the activation of a module is predicted from the landscape features, in our approach we take the opposite direction, i.e., we measure how much each feature contributes to the accuracy of the best regression models. We then perform an exploratory data analysis to investigate whether different features are needed to accurately predict the performance of the modular CMA-ES variants, and if these features can be linked to individual modules. Interestingly, we find that the *set* of features that are most relevant for an accurate regression does *not* depend on the configuration of the modules, indicating that feature selection does not necessarily need to be personalized to the specific configuration. The *relevance* of each feature, in contrast, can indeed be shown to depend on the characteristics of the algorithms. We show this by training a classifier that takes as input the vector that represents features' influence on the model accuracy (measured by approximated Shapley values in our analysis) and that outputs the status of individual modules of the CMA-ES configuration. For example, it is possible to predict whether or not a feature relevance vector stems from a configuration with *elitism* turned on or off. Interestingly, this can be predicted with good accuracy even when aggregating the feature relevance vectors across different budgets that the regression models were trained for, indicating that the impact of the landscape features is relatively stable for different stages of the optimization process.

Outline: Sec. 2 provides relevant background for our work. Sec. 3 presents the pipeline for our exploratory analysis. Key results are presented in Sec. 4, followed by a discussion of of our main findings and relevant steps for future work in Sec. 5.

Reproducibility: For our analysis, we trained a total number of 324 regression models for each of the 40 selected modular CMA-ES variants. We investigated fixed-budget performance for five different budgets and we considered two dimensions, $D = 5$ and $D = 30$. All performance data is available at [21]. Landscape data, code, and our results are available at [22].

2 BACKGROUND

In this section, we present the background of the two components that are parts of our proposed pipeline. We start by explaining exploratory landscape analysis (ELA), which is an approach for calculating numerical representations of the problem instances via features that describe their characteristics. In Sec. 2.2, we will then explain the process of learning a supervised machine learning model (i.e., an ELA-based performance regression model) that can be used to predict the quality of a solution achieved by the algorithm in a fixed-budget scenario.

2.1 ELA

The problem space covers optimization problems that can belong to the same or to different problem classes [4]. A problem class can have many different problem instances. The differences are a result of transformation processes such as translation, shifting, and/or scaling of the problem instance in the given problem space. These transformations can be stochastic in nature, defined as random variables, where input vectors are rotated, shifted, and/or scaled by predefined matrices and vectors. As a result, different problem instances of the same problem class can have different characteristics. To describe these properties of the problem instances, landscape analysis is applied. It calculates features by applying mathematical and statistical methods, which include different sampling methods and sample sizes. For example, exploratory landscape analysis (ELA), first introduced in [25], was designed to support the design of black-box optimization algorithms through a set of ML-based recommendations to obtain algorithms that best fit the problem at hand. The main objective of ELA is to capture the characteristics of optimization problems with a set of features, referred to as ELA features. These features are then used as input to the ML pipelines that produce the recommendations that guide the process of algorithm design. As we are dealing with black-box optimization, these features have to be calculated from an (ideally small) set of samples of the problem instance. The ELA features can be computed using the R-package *flacco* [19]. It contains 343 different ELA features grouped into 17 feature sets [18].

2.2 ELA-based performance regression model

The core motivation of creating ELA-based performance regression models lies in their ability to link the landscape features of the problem instance with the performance achieved by the algorithm on that problem instance. To collect the necessary training data for these models, we need to make use of a set of benchmark problem instances for which we have gathered performance data of our algorithms. This performance data can be viewed from two main perspectives: fixed-target (i.e., returns the budget required to achieve a certain target) and fixed-budget (i.e., estimates the quality of the solution achieved with a given computational budget). This performance data can then be linked to the ELA features collected on the specified benchmark function instances to train the regression models.

In this paper, we focus on the fixed-budget approach, where we predict the mean function value of the best solution that is reached by the algorithms after X function evaluations.

3 EXPLORATORY ANALYSIS PIPELINE

The proposed exploratory analysis pipeline consists of three main parts, i.e., learning a model for automated prediction of the algorithm's performance, feature importance calculation and ranking, which are further used to perform exploratory analysis and training

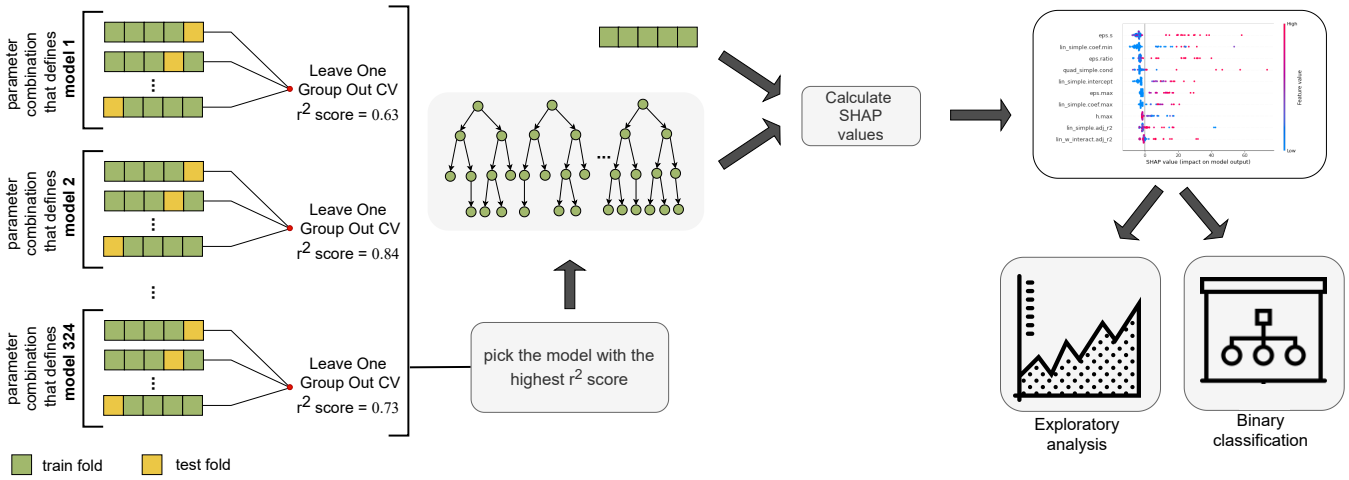


Figure 1: Overview of the exploratory analysis pipeline for linking problem landscape data with the performance of isolated modules of the CMA-ES algorithm.

classifiers to predict the status of a given CMA-ES module. The flowchart of the pipeline is presented in Fig. 1.

Regression model and its hyperparameters: To learn an algorithm performance prediction model, we took a supervised approach where each problem instance is described with a vector of n ELA features $(x_1, x_2, ..., x_n)$. These vectors are associated with one real value y indicating the performance of the algorithm on a specific problem instance after a fixed budget. Since the target that is being predicted is a real value, we are dealing with a regression task. Many studies have already tried to investigate this task in the context of automated algorithm performance prediction [27]. Here, we are learning Random Forest (RF) regression models where we also considered tuning part of the hyperparameters. Previous studies have already shown that RF can provide promising results dealing with automated performance prediction [17]. For hyperparameter tuning, we employed the grid search methodology, which performs an exhaustive search over a manually selected finite subset of candidate solutions of the hyperparameter space of the algorithm.

Leave-One-Group-Out validation and selection of the best model: For validation of the induced models, we use the Leave-One-Group-Out strategy. Let us assume that for each problem, m instances are involved. In our case, groups are denoted by the problem instances, which means that we ended up with m groups. The evaluation metric of choice is the r^2 score. The same evaluation metric is used for evaluating the individual model candidates and as a heuristic in the grid search step, where based on the r^2 score we identify the best performing model (the model with the highest r^2 score).

Calculating the explanations of ML regression model: Once the predictive regression model is learned, we then proceed with the calculation of the Shapley values, i.e. the ELA feature importance scores. The Shapely values are quantification of the marginal contribution of each input feature on the predictions made by the predictive model [26]. The computation of the Shapley values is computationally expensive as it considers each possible combination of features (a power-set of features) to determine their importance on the prediction. We therefore apply the SHAP (SHaply Additive exPlanations) algorithm [23]. The SHAP algorithm was first introduced in 2017 as a game-theoretic approach to explaining the output of ML models. Since then, it has successfully been applied to reverse-engineer the output of predictive algorithms (including the so-called black-box algorithms).

Exploratory analysis: The pipeline for learning a regression model and obtaining the SHAP values to explain its predictions is repeated for each algorithm configuration considered in the study (explained in Sec. 4.1.2). The algorithm configurations are then grouped with respect to a given module to observe whether there are some differences in the performance when there are some specific structural changes to the algorithm. For simplicity of the analysis, the groups of algorithm configurations are set to always differ in one module of the CMA-ES algorithm. For example, in the modular implementation of the CMA-ES algorithm, the elitism module can be either turned on or off, while the step-size-adaptation module, if active, can take 2 different values, i.e., csa and psr. Such an approach facilitates the exploratory analysis of the effect each of these models has on the final performance of the algorithm. Furthermore, trends in the landscape space can be observed. More specifically, some ELA features can be found to hold more predictive value than the rest by observing the SHAP values across the different problems in multiple dimensions and different thresholds for the maximum number of function evaluation runs.

Classifiers for predicting the status of a CMA-ES module: Using the calculated Shapley values aggregated across all benchmark problem instances, we can generate a vector representation for each configuration. This representation is a vector of Shapley values, one per each ELA feature, which convey the information of how the ELA feature contributes to the prediction of the performance of the configuration. Further, these representations can be labeled with regard to the activation status of a module in modular CMA-ES (e.g.,

taking into account the elitism, the labels will be “ON" and “OFF"). Using the annotated data, we can learn classifiers that can predict from this representation the status of a given CMA-ES module.

4 RESULTS & DISCUSSION

This section describes the experimental setup, which includes the description of landscape and performance data, as well as the details of hyperparameter tuning used in the grid search for the RF models. Following that, we will present the analysis in which we examine the effect of elitism and step-size modules on modular CMA-ES performance.

4.1 Experimental setup

4.1.1 Landscape data. The problem portfolio consists of the first five instances of the 24 noiseless BBOB functions [14] of the COCO benchmark environment [13] in both $D = 5$ and $D = 30$ dimensions. Thus, we obtain two problem sets (one for each dimension) containing 120 problem instances each. The problem landscape data are represented using 46 of the so-called "cheap" ELA features implemented in the R package `flacco` [19]. Here, however, we do not calculate the ELA features but instead, make use of a publicly available dataset that contains the necessary landscape data [30]. The ELA features utilized in the study can be divided into 6 groups, i.e., dispersion, y-distribution, meta model, information content, nearest better clustering, and principal component analysis. The selected ELA features have been calculated using the Sobol sampling strategy with $100D$ sample size on a total of 100 independent repetitions. For a more robust analysis, we use the median of the 100 calculated feature values. This data is also made easily accessible via the OPTION ontology [20] and we used their API to extract it.

4.1.2 Performance data. Since collecting data based on a complete enumeration of all combinations of modular CMA-ES modules is computationally infeasible, we make use of a set of 40 configurations. The details of which ones were selected and used can be found in our Zenodo repository [21]. For each of these configurations, we collect 10 independent runs on each of the first 5 instances of the BBOB functions, in both 5 and 30 dimensions. For each of these runs, we then extract the best precision reached after $B = \{500, 2\,000, 5\,000, 10\,000, 50\,000\}$ function evaluations.

4.1.3 RF hyperparameter tuning. For learning the RF models for automated performance prediction, we used the Python package scikit-learn [28]. The hyperparameters of the RF model are estimated with the Grid Search strategy. We considered tuning 5 different RF parameters: (1) n_estimators - the number of trees (estimators) in the forest; (2) max_features - the number of features considered when making the best split ; (3) max_depth - the maximum depth of the tree; (4) min_samples_split - the minimum number of samples required for splitting a node in the tree; and (5) criterion - the function that measures the quality of a split. The values for the aforementioned hyperparameter used in the grid search are presented in Tab. 1. This resulted in training 324 model candidates for each modular CMA-ES algorithm configuration involved in the study. Note that we have 40 different algorithm configurations. However, since we consider problem instances in 2 different dimensions ($5D$ and $30D$) and we generate performance data for 5 budgets, we essentially end up learning 400 ($40 \times 2 \times 5$) predictive models. Hence a total of 129 600 (324×400) RF models have been trained for the purpose of this study. The best model for each of the 400 configuration was selected using the r^2 evaluation metric.

Table 1: RF hyperparameters and their values used in the grid search.

hyperparameter	search space
n_estimators	$[100, 500, 1000]$
max_features	$['auto', 'sqrt', 'log2']$
max_depth	$[4, 8, 15, None]$
min_samples_split	$[2, 5, 10]$
criterion	$['squared\_error', 'absolute\_error', 'poisson']$

4.2 Impact of the elitism module

To estimate the impact of the elitism module on the modular CMA-ES performance, we used the proposed exploratory analysis pipeline. For this purpose, from the 40 algorithm configurations, we identified 11 pairs of configurations in which all the modules are set the same, but the elitism module is either activated or not. An example of such a pair is depicted in Tab. 2.

As mentioned previously in the exploratory analysis pipeline, for each algorithm configuration from the 11 pairs (total of 22 configurations) a separate RF with the optimal hyperparameters is learned. Following that, the SHAP algorithm was used to determine the feature importance for each of the 46 ELA features. To provide global explanations about the impact of elitism on algorithm performance, we jointly analyze the 11 pairs.

We are using the leave-one-group-out validation with 5 groups (four for training and one for testing). However, we are interested in looking at the Shapley values for the training folds only as this is the data that is used for learning the predictive models and can provide explanations on how the model works. Hence, the validation involves calculating the Shapley values four times for each benchmark problem instance (each problem instance is represented four times in the training data and once in the testing data). To obtain a single Shapley value for each ELA feature and problem instance, the mean of the four values is calculated. Additionally, to select the most important ELA features across all benchmark problem instances, we calculate the mean of the Shapley values across all problem instances. Finally, we select the top k most important features to perform the exploratory analysis.

First, we have conducted an analysis using the top 10 most important ELA features. Then, we count the number of times a given feature appears in the top 10 across all pairs of problem instances in 5 and 30 dimensions and with 5 different budgets. In other words, we calculate the frequency with which a given ELA feature appears in the top 10 most important as indicated by their Shapley values. This value ranges between 0 and the total number of configuration pairs considered (in our case 11). A value of zero indicates that an ELA feature was not ranked in the top 10 in any of the 11 configuration pairs, whereas a value of 11 indicates that the feature was ranked in the top 10 in all 11 configurations. The results are shown

Table 2: An example pair of algorithm configurations that differ only in one of the modules of the CMA-ES algorithm, i.e., in the elitism module (active and non-active).

elitist	mirrored	base_sampler	weights_option	local_restart	bound_correction	step_size_adaptation
True	mirrored	gaussian	default	OFF	saturate	csa
False	mirrored	gaussian	default	OFF	saturate	csa

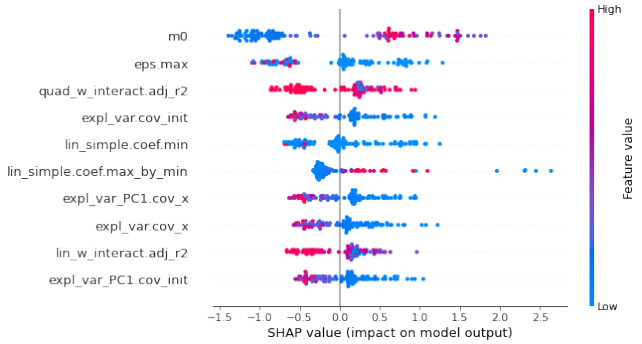
		5D										30D									
ELA feature	sum	0-500	1-500	0-2000	1-2000	0-5000	1-5000	0-10000	1-10000	0-50000	1-50000	0-500	1-500	0-2000	1-2000	0-5000	1-5000	0-10000	1-10000	0-50000	1-50000
diff_median_02	12	0	0	0	0	0	0	0	0	0	0	6	6	0	0	0	0	0	0	0	0
dist_ratio.coeff_var	11	0	0	6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0
eps.max	214	11	10	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	9	8
eps.ratio	171	11	11	1	8	7	9	9	11	7	7	11	11	11	11	11	11	11	11	0	2
eps.s	119	9	10	0	0	0	1	0	0	1	1	11	11	11	11	11	11	11	11	2	7
expl_var.cor_x	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
expl_var.cov_init	125	0	1	11	11	11	11	11	11	11	11	0	0	0	0	0	0	4	10	11	11
expl_var.cov_x	89	2	3	9	11	9	10	8	7	7	4	0	0	0	0	0	0	0	0	11	8
expl_var_PC1.cov_init	82	0	0	9	8	10	6	9	5	8	6	0	0	0	0	0	0	0	0	11	10
expl_var_PC1.cov_x	84	0	0	9	10	10	8	9	9	6	10	0	0	0	0	0	0	0	0	10	3
h.max	63	8	5	0	2	0	3	0	0	0	0	5	5	3	8	5	5	10	4	0	0
kurtosis	24	0	0	0	0	0	0	0	0	0	0	11	11	0	2	0	0	0	0	0	0
lin_simple.adj_r2	31	6	5	0	0	0	0	0	0	0	0	0	0	7	3	5	5	0	0	0	0
lin_simple.coef.max	123	11	10	0	0	1	2	1	0	1	0	11	11	11	11	11	11	11	11	2	7
lin_simple.coef.max_by_min	113	6	4	9	10	8	8	7	9	11	11	0	0	1	0	1	1	2	6	9	10
lin_simple.coef.min	215	9	10	9	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
lin_simple.intercept	191	11	11	5	6	7	7	10	8	9	9	11	11	11	11	11	11	11	11	9	11
lin_w_interact.adj_r2	44	6	6	2	2	4	3	4	6	6	5	0	0	0	0	0	0	0	0	0	0
m0	103	0	0	10	11	11	11	11	11	11	11	0	0	0	0	0	0	0	0	9	7
nb_fitness.cor	34	2	4	1	0	0	0	0	0	1	0	11	11	0	4	0	0	0	0	0	0
nn_nb.cor	29	2	2	7	0	3	0	3	0	1	1	0	0	1	0	1	1	1	0	4	2
nn_nb.mean_ratio	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nn_nb.sd_ratio	24	4	4	1	0	0	0	0	0	0	0	0	0	2	3	4	4	2	0	0	0
number_of_peaks	10	0	3	0	0	0	0	0	0	0	0	0	0	2	1	2	2	0	0	0	0
quad_simple.adj_r2	71	1	0	0	0	0	0	0	0	1	1	0	0	7	3	8	8	11	11	10	10
quad_simple.cond	90	8	11	1	0	0	0	0	0	1	0	0	0	11	11	11	11	11	11	0	3
quad_w_interact.adj_r2	99	2	0	6	9	6	9	6	11	6	11	0	0	10	6	7	7	3	0	0	0
skewness	26	0	0	1	0	0	0	0	0	0	0	11	11	0	3	0	0	0	0	0	0

Figure 2: Frequency of appearance of the ELA features as one of the top 10 most important features in the 11 pairs of configurations of modular CMA-ES with elitism turned off and on. The results are calculated on the 24 BBOB functions in both 5 and 30 dimensions and where the best precision is reached after $B = \{500, 2\,000, 5\,000, 10\,000, 50\,000\}$ function evaluations.

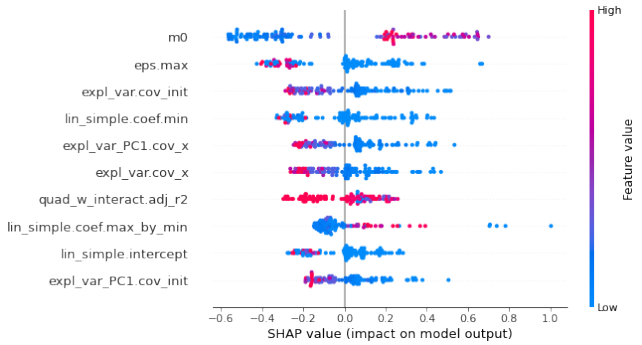
in Fig. 2. Note that due to page limitation, in Fig. 2 we only list the features that appear in the top 10 at least once.

Focusing on Fig. 2 and $D = 5$, we can see that several ELA features are unaffected by the status of the elitism module (whether the elitism module is active or not), and even appear in the top 10 most important features regardless of the budget in which the algorithm performance is predicted. These features include “eps.max", “lin_simple.coef.max_by_min", and “lin_simple.coef.min". There are several features, such as “eps.s", “h.max", “lin_simple.coev.max", and “quad_simple.cond" that seem to be important when predicting the performance for small budgets (500 function evaluations) both when elitism is active and not, but are not important for larger budgets. The results indicate that these features help predict the performance when the algorithm is still in the exploration phase, looking for areas in search space with a high probability of finding good solutions. The opposite is true for another group of features (“expl_var.cor_x", “expl_var.cov_x", “expl_var_PC1.cov_init", “expl_var_PC1.cov_x", and “m0"). They are not important for small budgets, however, when the budget increases (≥1000 function evaluations), they become important regardless of whether the elitism module is on or off. The most interesting ELA feature from this analysis is “quad_w_interact.adj_r2". At the beginning of the optimization process (when the budget is set to 500 function evaluations), it is not important whether elitism is on or off, however as the budget increases, it is visible that this feature is more important in the cases when elitism is on. We can conclude that the majority of the ELA features are behaving similarly regardless of the elitism module activation state. From the results, it follows that there is no special group of features that are related to the elitism module. On the other hand, this provides promising results that the same ELA feature portfolio can be enough to predict the performance of different modular CMA-ES no matter the status of the elitism in the algorithm’s configuration.

After summarizing the importance of the ELA features globally, we can use the proposed exploratory analysis pipeline to provide local explanations for a pair of configurations that differ only in the elitism module. Here, we will discuss the local explanations for the pair presented in Tab. 2. Fig. 3 presents the summary Shapley plots for both configurations in the pair separately. These summary plots illustrate the positive and negative relationships with the quality of solution reached after some fixed budget. Each dot in the plots represents a problem instance. The ELA features are listed in descending order of importance. The colors used indicate the magnitude of the ELA feature value (red representing higher values



(a) Elitism activated



(b) Elitism not activated

Figure 3: Shapley values of the top 10 ELA features for a pair of configuration presented in Tab. 2, for $5D$ and 2000 function evaluations.

and blue representing lower values). Finally, the position on the horizontal axis presents the impact of the ELA feature value on the prediction of the target. We observe that regardless of whether the elitism module is activated, the top 10 most important ELA features are mostly the same, and their contributions of the prediction to the algorithm performance follow similar patterns.

Focusing on $D = 30$ (Fig. 2), similar patterns as in $D = 5$ can be observed. However, we will not compare them across different dimensions since for the simplicity of analysis the selected budgets are the same and a larger budget is likely required to solve the problem instances in high dimension. Here, "eps.max", "eps.ratio", "eps.s", "lin_simple.coef.max", "lin_simple.coef.min", and "lin_simple.coef.intercept", are important for various budgets regardless of the elitism activation status. The "expl_var.cov_init", "expl_var.cov_x", "expl_var_PC1.cov_init", "expl_var_PC1.cov_x", "m0" are important only for large budgets (50 000), again, regardless of the elitism activation status. The statistical features "diff_median_02", "kurtosis", and "skewness" are important only for small budgets when the dimension is 30.

To determine whether these findings hold true as the number of the most important ELA features considered increases, Fig. 4 presents the same analysis performed for $5D$ and $30D$ for all 5 budgets when the top 20 features are selected instead of the top 10. Without going into detail, we can conclude that similar patterns exist for the majority of the previously described features. The difference is that more features appear, which is logical given the selection of 20 features. Certain patterns observed in the top 10 selection do change. For instance, when focusing on $5D$, "eps.s" and "lin_simple.coef.max" appear to be important also for larger budgets, which was not the case in top 10 features analysis, where they appeared only for really small budgets. It is also interesting that the feature, "quad_w_interact.adj_r2", that was more important when the elitism was on, now with the top 20 analysis seems to be equally important regardless of the activation of the elitism module.

The exploratory analysis provides us with some descriptive analysis using a counting procedure. To see if we can detect differences between the configurations when the elitism module is switched on or off, we represented each configuration as a vector of 46 Shapley values for each budget separately and then we labeled the vector with "ON" or "OFF" with regard to the elitism activation status. Here, we ended up with 110 (11 pairs $\times$ 2 configurations $\times$ 5 budgets) configurations (i.e., data instances) represented with 46 Shapley values that provide representations of how the ELA features are utilized to predict each configuration performance. The distribution of the labels within the 110 configurations is 50:50, so the baseline of classifying them with regard to the activation of the elitism is 50%. The process of labeling the data was done twice, so we ended up with two datasets, one for each dimension.

Next, for each dataset separately, we trained a binary classifier that uses the Shapley representations as input data and predicts if the elitism is ON or OFF. We utilized RF for learning the classifiers and we evaluated them in 5 cross-fold validation using the 110 instances. We used RF with default parameters as implemented in the Python package scikit-learn.

Tab. 3a presents the accuracy and the F_1 score obtained for the classification done in each problem dimension. In $5D$, the classifier provides 79% accuracy of classifying the status of elitism (ON or OFF), while in $30D$ the accuracy is around 66%. It is important to note that these results are obtained regardless of the budget value. In both dimensions, the accuracy is above the baseline, which shows promising results. These results indicate that an automated performance prediction model built using the explanations of training contains useful information that can be used to predict whether a configuration has the elitism module activated or not. Although the importance of the ELA features is largely the same regardless of the activation status of the module, these findings show that their contribution (aggregated Shapley values across all problem instances) is different, allowing us to determine whether a module is active or not.

4.3 Impact of the step-size module

In our second use case, we used the exploratory analysis pipeline to investigate the impact of the step-size-adaptation module that can be switched between two values: csa and psr. Here, 18 pairs of configurations that differ only in the step-size adaptation module have been identified and selected for further analysis. Fig. 5 presents the results of the exploratory analysis performed with the selection of the top 10 most important ELA features. As a result, there appears to be no clear difference between the two possible values of this

		5D										30D									
ELA feature	sum	0-500	1-500	0-2000	1-2000	0-5000	1-5000	0-10000	1-10000	0-50000	1-50000	0-500	1-500	0-2000	1-2000	0-5000	1-5000	0-10000	1-10000	0-50000	1-50000
diff_mean_02	26	1	2	2	0	0	0	0	0	0	0	4	3	1	6	2	3	1	0	1	0
diff_mean_05	46	0	0	2	5	4	5	5	8	5	3	1	4	0	0	0	0	0	0	2	2
diff_mean_10	14	2	0	0	0	0	0	0	0	0	0	4	5	0	0	0	0	0	0	1	2
diff_median_02	42	1	4	1	0	0	0	0	0	0	0	9	9	3	7	3	3	2	0	0	0
diff_median_05	64	0	0	7	3	6	6	6	4	3	5	10	9	0	0	0	0	0	0	3	2
dist_ratio.coeff_var	160	3	6	9	11	7	11	7	11	5	11	0	0	10	11	9	11	11	11	9	7
eps.max	220	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
eps.ratio	219	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	10
eps.s	217	11	11	9	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	10	11
expl_var.cor_init	13	0	0	4	0	2	0	2	1	2	1	0	1	0	0	0	0	0	0	0	0
expl_var.cor_x	40	4	1	7	3	5	3	5	3	2	4	2	1	0	0	0	0	0	0	0	0
expl_var.cov_init	155	3	8	11	11	11	11	11	11	11	11	7	7	1	0	1	0	7	11	11	11
expl_var.cov_x	147	8	11	11	11	11	11	11	11	11	11	0	0	1	0	1	2	4	10	11	11
expl_var_PC1.cor_init	4	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
expl_var_PC1.cor_x	3	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0
expl_var_PC1.cov_init	138	5	3	11	11	11	11	11	11	11	11	0	1	0	0	0	0	8	11	11	11
expl_var_PC1.cov_x	137	2	7	11	11	11	11	11	11	11	11	2	5	0	0	1	0	4	6	11	11
h.max	150	9	11	2	9	1	5	1	6	5	5	11	11	11	11	11	11	11	11	1	7
kurtosis	68	6	7	0	0	0	0	0	0	0	0	11	11	5	9	8	7	3	1	0	0
lin_simple.adj_r2	99	11	11	1	1	2	1	3	0	0	0	3	2	11	11	11	11	6	6	4	4
lin_simple.coef.max	216	11	11	8	11	11	11	11	11	10	11	11	11	11	11	11	11	11	11	11	11
lin_simple.coef.max_by_min	194	11	11	11	11	11	11	11	11	11	11	3	2	9	10	7	10	10	11	11	11
lin_simple.coef.min	220	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
lin_simple.intercept	217	11	11	8	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
lin_w_interact.adj_r2	154	11	9	8	11	10	11	11	11	11	11	0	2	4	2	10	8	3	0	11	10
m0	144	0	5	11	11	11	11	11	11	11	11	8	8	0	3	0	0	2	8	11	11
nb_fitness.cor	74	10	6	4	1	4	2	4	0	4	0	11	11	8	5	1	1	1	0	0	1
nn_nb.cor	182	7	7	11	11	10	11	10	11	8	11	1	1	10	7	11	11	11	11	11	11
nn_nb.mean_ratio	84	5	6	4	0	0	0	0	0	0	0	1	2	11	11	11	10	11	8	3	1
nn_nb.sd_ratio	98	9	8	2	0	0	0	0	0	0	1	2	1	10	9	11	11	11	11	7	5
number_of_peaks	98	10	9	0	0	0	0	0	0	1	0	8	5	11	11	11	11	11	10	0	0
quad_simple.adj_r2	126	10	4	0	2	2	3	1	3	7	6	2	4	11	6	10	11	11	11	11	11
quad_simple.cond	212	11	11	10	11	11	11	11	11	11	11	9	7	11	11	11	11	11	11	10	11
quad_w_interact.adj_r2	170	7	0	8	11	10	11	11	11	11	11	3	4	11	6	11	11	9	6	9	9
ratio_mean_02	30	0	1	1	0	0	0	0	0	0	0	6	5	4	4	3	4	2	0	0	0
ratio_mean_05	34	0	0	1	2	2	3	4	2	5	5	4	2	0	0	0	0	0	0	2	2
ratio_mean_10	6	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	1	2
ratio_mean_25	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ratio_median_02	49	2	4	0	0	0	0	0	0	0	0	10	9	8	5	5	3	2	0	1	0
ratio_median_05	62	0	2	8	7	8	5	4	6	3	3	6	7	0	0	0	0	0	0	2	1
ratio_median_10	4	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	1
ratio_median_25	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
skewness	62	2	0	3	0	3	0	3	0	6	0	11	11	3	9	5	4	2	0	0	0

Figure 4: Frequency of appearance of the ELA features as one of the top 20 most important features in the 11 pairs of configurations of modular CMA-ES with elitism turned off and on. The results are calculated on the 24 BBOB functions in both 5 and 30 dimensions and where the best precision is reached after $B = \{500, 2\,000, 5\,000, 10\,000, 50\,000\}$ function evaluations.

	Problem dimension	
	$5D$	$30D$
Accuracy	0.79090	0.66364
F_1 score	0.78938	0.66225

(a) Activation of the elitism module

	Problem dimension	
	$5D$	$30D$
Accuracy	0.76111	0.68333
F_1 score	0.75523	0.68149

(b) Step-size-adaptation set to csa and psr

Table 3: Binary classification results.

module, and the same features appear to contribute the same in both cases. Again, there are features that are equally important for all budgets, features that are important at the beginning of the optimization process but are less important for larger budgets, and features that are not important at the beginning of the optimization process but are important for larger budgets, regardless of the value of the step-size-adaptation module.

To go beyond the descriptive analysis, we have again trained classifiers for predicting the step-size adaptation (csa or psr) by using the Shapley representations of the configuration. Here, we ended up with 180 (18 pairs × 2 configurations × 5 budgets) configurations (data instances) labeled with "CSA" or "PSR". The classifier baseline is 50%, the same as for the elitism module. For each dimension, we learned an RF model for predicting the step-size-adaptation value. We used the same experimental setting as in the classification analysis for the elitism module (RF with default parameters and 5-fold cross-validation). The results are presented in Tab. 3b. For both dimensions, the accuracy is above the baseline, 76%, and 68% for $5D$ and $30D$, respectively. These results point out that the Shapley representations of the configurations that consist of the information of how the ELA features contribute to predicting their performance are useful to predict the value of the step-size-adaption module.

		5D										30D									
ELA feature	sum	0-500	1-500	0-2000	1-2000	0-5000	1-5000	0-10000	1-10000	0-50000	1-50000	0-500	1-500	0-2000	1-2000	0-5000	1-5000	0-10000	1-10000	0-50000	1-50000
diff_median_02	18	0	0	0	0	0	0	0	0	0	0	11	7	0	0	0	0	0	0	0	0
dist_ratio.coeff_var	14	0	0	5	2	1	0	0	0	0	0	0	0	0	0	0	0	0	4	2	0
eps.max	354	18	17	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	14	17
eps.ratio	269	18	18	7	11	11	12	13	14	8	9	18	18	18	18	18	18	18	18	0	4
eps.s	194	18	14	0	0	1	1	0	1	1	1	18	18	18	18	18	18	18	18	4	9
expl_var.cor_x	2	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
expl_var.cov_init	206	1	0	18	18	18	18	18	18	18	18	1	1	0	0	0	0	10	13	18	18
expl_var.cov_x	160	4	5	17	17	17	16	18	11	15	10	0	0	0	0	0	0	0	0	15	15
expl_var_PC1.cov_init	149	0	0	14	15	14	16	15	14	13	13	0	0	0	0	0	0	0	1	18	16
expl_var_PC1.cov_x	139	0	0	15	17	14	15	14	17	13	16	0	0	0	0	0	0	0	0	11	7
h.max	108	11	11	5	0	4	0	0	1	0	0	6	9	13	9	8	8	16	6	1	0
kurtosis	41	0	0	0	0	0	0	0	0	0	0	18	18	5	0	0	0	0	0	0	0
lin_simple.adj_r2	46	9	10	0	0	0	0	0	0	0	0	0	0	4	8	8	7	0	0	0	0
lin_simple.coef.max	202	18	17	0	0	2	1	1	0	1	0	18	18	18	18	18	18	18	18	7	11
lin_simple.coef.max_by_min	196	13	5	16	17	15	14	14	15	18	18	0	0	0	1	1	2	10	5	17	15
lin_simple.coef.min	352	15	16	16	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	17	18
lin_simple.intercept	297	18	18	4	11	7	16	9	15	9	14	18	18	18	18	18	18	18	18	16	16
lin_w_interact.adj_r2	61	10	7	5	2	4	5	7	5	11	5	0	0	0	0	0	0	0	0	0	0
m0	173	0	1	17	18	18	18	18	18	18	18	0	0	0	0	0	0	0	1	13	15
nb_fitness.cor	56	5	1	0	1	1	0	0	0	0	1	18	18	11	0	0	0	0	0	0	0
nn_nb.cor	37	0	6	5	3	3	1	3	0	1	1	0	0	0	1	1	3	1	1	6	1
nn_nb.mean_ratio	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nn_nb.sd_ratio	34	1	12	1	0	0	0	0	0	0	0	0	0	2	4	5	7	0	2	0	0
number_of_peaks	16	4	1	0	0	0	0	0	0	0	0	0	0	5	0	3	3	0	0	0	0
quad_simple.adj_r2	125	0	3	0	0	0	2	0	2	0	7	0	0	2	14	15	13	17	18	18	14
quad_simple.cond	151	16	16	0	1	0	0	0	0	1	2	0	0	18	18	18	18	18	18	3	4
quad_w_interact.adj_r2	153	1	1	14	11	14	9	14	13	16	11	0	0	5	17	13	11	0	3	0	0
ratio_median_05	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
skewness	44	0	0	1	0	0	0	0	0	0	0	18	18	7	0	0	0	0	0	0	0

Figure 5: Frequency of appearance of the ELA features as one of the top 10 most important features in the 18 pairs of configurations of modular CMA-ES with step-size-adaptation module set to csa and psr. The results are calculated on the 24 BBOB functions in both 5 and 30 dimensions and where the best precision is reached after $B = \{500, 2\,000, 5\,000, 10\,000, 50\,000\}$ function evaluations.

5 CONCLUSIONS AND FUTURE WORK

In this study, we have investigated the impact of the landscape features on the performance of two modules involved in modular CMA-ES, elitism, and step-size-adaptation. The same pipeline can be applied to the other modules as well. The explanations (Shapley values for each ELA feature) from automated algorithm performance prediction are used as vector representations for each modular CMA-ES. For both investigated modules (elitism and step-size adaptation), it seems that the same ELA features are within the top 10, 15, and 20 most important features that contribute to their performance prediction (using their ranking) regardless of the possible values of each module. Hence, the main take away message is that we can do feature selection for all algorithms at once, no matter the configuration of the algorithm. However, it is advisable to train separate regression models for each configuration.

The study has been conducted on a limited number of modular CMA-ES configurations. With more training data we would expect the classifiers' performance to improve even more. The presented pipeline can directly be used to analyze other CMA-ES modules not considered in this work.

Such supervised classifiers can further be used to predict the modules for which a random modular CMA-ES is composed. Moreover, having performance data from some other algorithms such as DE or PSO obtained on the same problem instances will allow us to run the pipeline and train an explainable regression model, whose explanations will further be used as their Shapley representations. Finally by using classifiers that can predict the status of each module in the modular CMA-ES, we can use the learned representations for DE or PSO with each classifier, to be able to find which modules should be activated and with which values in order to find similar modular CMA-ES configuration (to mimic the behavior of the other algorithms through the modular CMA-ES). This is a direction we plan to explore in our future work.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Slovenian Research Agency through research core grants No. P2-0103 and P2-0098, project grants No. J2-9230 and N2-0239, and the young researcher grant No. PR-09773 to AK, as well as the EC through grant No. 952215 (TAILOR). Our work is also supported by Paris Ile-de-France region, via the DIM RFSI AlgoSelect project and via a SPECIES scholarship for Ana Kostovska.

REFERENCES

[1] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. 2011. Mirrored Sampling in Evolution Strategies with Weighted Recombination. In *GECCO*. ACM, 861–868. https://doi.org/10.1145/2001576.2001694

[2] Anne Auger and Nikolaus Hansen. 2005. A restart CMA evolution strategy with increasing population size. In *Proc. of Congress on Evolutionary Computation (CEC'05)*. 1769–1776. https://doi.org/10.1109/CEC.2005.1554902

[3] Amine Aziz-Alaoui, Carola Doerr, and Johann Dréo. 2021. Towards large scale automated algorithm design by integrating modular benchmarking frameworks. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21, Companion material)*. ACM, 1365–1374.

[4] Thomas Bartz-Beielstein, Carola Doerr, Daan van den Berg, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, William La Cava, Manuel Lopez-Ibanez, et al. 2020. Benchmarking in optimization: Best practice and open issues. *arXiv preprint arXiv:2007.03488* (2020).

[5] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer. 2017. Per instance algorithm configuration of CMA-ES with limited budget. In *Proc. of Genetic and Evolutionary Computation (GECCO'17)*. ACM, 681–688. https://doi.org/10.1145/3071178.3071343

[6] Rick Boks, Hao Wang, and Thomas Bäck. 2020. A modular hybridization of particle swarm optimization and differential evolution. In *GECCO '20: Genetic and Evolutionary Computation Conference, Companion Volume, Cancún, Mexico, July 8-12, 2020*, Carlos Artemio Coello Coello (Ed.). ACM, 1418–1425. https://doi.org/10.1145/3377929.3398123

[7] C.L. Camacho Villalón, M. Dorigo, and T. Stützle. 2021. *PSO-X: A component-based framework for the automatic design of particle swarm optimization algorithms.* Technical Report TR/IRIDIA/2021-002. IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

[8] Jacob de Nobel, Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. 2021. Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21, Companion material)*, Krzysztof Krawiec (Ed.). ACM, 1375–1384. https://doi.org/10.1145/3449726.3463167

[9] Jacob de Nobel, Hao Wang, and Thomas Baeck. 2021. Explorative data analysis of time series based algorithm features of CMA-ES variants. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 510–518.

[10] Alberto Franzin and Thomas Stützle. 2019. Revisiting simulated annealing: A component-based analysis. *Comput. Oper. Res.* 104 (2019), 191–206. https://doi.org/10.1016/j.cor.2018.12.015

[11] Nikolaus Hansen. 2009. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers* (Montreal, Québec, Canada) *(GECCO '09)*. Association for Computing Machinery, New York, NY, USA, 2389–2396. https://doi.org/10.1145/1570256.1570333

[12] Nikolaus Hansen. 2019. A global surrogate assisted CMA-ES. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 664–672.

[13] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. 2020. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software* (2020), 1–31.

[14] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions.* Research Report RR-6829. INRIA. https://hal.inria.fr/inria-00362633

[15] Nikolaus Hansen and Andreas Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proc. of IEEE Congress on Evolutionary Computation (CEC'96)*. IEEE, 312–317. https://doi.org/10.1109/ICEC.1996.542381

[16] Anja Jankovic and Carola Doerr. 2020. Landscape-aware fixed-budget performance regression and algorithm selection for modular CMA-ES variants. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 841–849.

[17] Anja Jankovic, Gorjan Popovski, Tome Eftimov, and Carola Doerr. 2021. The Impact of Hyper-Parameter Tuning for Landscape-Aware Performance Regression and Algorithm Selection. In *Genetic and Evolutionary Computation Conference (GECCO 2021)*.

[18] P. Kerschke, H.H. Hoos, F. Neumann, and H. Trautmann. 2019. Automated Algorithm Selection: Survey and Perspectives. *Evolutionary Computation* 27, 1 (March 2019), 3–45.

[19] P. Kerschke and H. Trautmann. 2016. The R-Package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In *CEC*. 5262–5269. https://doi.org/10.1109/CEC.2016.7748359

[20] Ana Kostovska, Diederick Vermetten, Carola Doerr, Saso Dzeroski, Panče Panov, and Tome Eftimov. 2021. OPTION: OPTImization Algorithm Benchmarking ONtology. In *Genetic and Evolutionary Computation Conference (GECCO 2021, Companion Material)*.

[21] Ana Kostovska, Diederick Vermetten, Sašo Džeroski, Carola Doerr, Peter Korošec, and Tome Eftimov. 2022. *Linking Problem Landscape Features with the Performance of Individual CMA-ES Modules - Data.* https://doi.org/10.5281/zenodo.5947076

[22] Ana Kostovska, Diederick Vermetten, Sašo Džeroski, Carola Doerr, Peter Korošec, and Tome Eftimov. 2022. *Linking Problem Landscape Features with the Performance of Individual CMA-ES Modules - Data and Code.* https://github.com/KostovskaAna/LinkingELAToPerformance

[23] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[24] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *Proc. of Genetic and Evolutionary Computation (GECCO'11)*. ACM, 829–836.

[25] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. 2011. Exploratory Landscape Analysis. In *Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 829–836.

[26] Christoph Molnar. 2020. *Interpretable machine learning.* Lulu. com.

[27] Mario A Muñoz, Michael Kirley, and Saman K Halgamuge. 2012. A meta-learning prediction model of algorithm performance for continuous optimization problems. In *International Conference on Parallel Problem Solving from Nature*. Springer, 226–235.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[29] Raphael Patrick Prager, Heike Trautmann, Hao Wang, Thomas HW Bäck, and Pascal Kerschke. 2020. Per-Instance Configuration of the Modularized CMA-ES by Means of Classifier Chains and Exploratory Landscape Analysis. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 996–1003.

[30] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. 2020. *Experimental Data Set for the study "Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy".* https://doi.org/10.5281/zenodo.3886816

[31] Risto Trajanov, Stefan Dimeski, Martin Popovski, Peter Korošec, and Tome Eftimov. 2021. Explainable Landscape-Aware Optimization Performance Prediction. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 01–08.

[32] Sander Van Rijn, Carola Doerr, and Thomas Bäck. 2018. Towards an adaptive CMA-ES configurator. In *International Conference on Parallel Problem Solving from Nature*. Springer, 54–65.

[33] Sander van Rijn, Hao Wang, Matthijs van Leeuwen, and Thomas Bäck. 2016. Evolving the structure of Evolution Strategies. In *Proc. of IEEE Symposium Series on Computational Intelligence (SSCI'16)*. IEEE, 1–8. https://doi.org/10.1109/SSCI.2016.7850138

[34] Sander van Rijn, Hao Wang, Bas van Stein, and Thomas Bäck. 2017. Algorithm Configuration Data Mining for CMA Evolution Strategies. In *GECCO*. ACM, 737–744. https://doi.org/10.1145/3071178.3071205

[35] Diederick Vermetten, Sander van Rijn, Thomas Bäck, and Carola Doerr. 2019. Online selection of CMA-ES variants. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 951–959.

[36] Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Back. 2020. Integrated vs. sequential approaches for selecting and tuning CMA-ES variants. In *ACM Genetic and Evolutionary Computation Conference (GECCO'20)*.

[37] Hao Wang, Michael Emmerich, and Thomas Bäck. 2014. Mirrored Orthogonal Sampling with Pairwise Selection in Evolution Strategies. In *SAC*. ACM, 154–156. https://doi.org/10.1145/2554850.2555089

[38] Furong Ye, Hao Wang, Carola Doerr, and Thomas Bäck. 2020. Benchmarking a (μ +λ) Genetic Algorithm with Configurable Crossover Probability. In *Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12270)*, Thomas Bäck, Mike Preuss, André H. Deutz, Hao Wang, Carola Doerr, Michael T. M. Emmerich, and Heike Trautmann (Eds.). Springer, 699–713. https://doi.org/10.1007/978-3-030-58115-2_49