Towards Feature-Based Performance Regression Using Trajectory Data

Anja Jankovic, Tome Eftimov, Carola Doerr

Towards Feature-Based Performance Regression Using Trajectory Data

Anja Jankovic[1], Tome Eftimov[2], and Carola Doerr[1]

[1]Sorbonne Université, CNRS, LIP6, Paris, France
[2]Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia

Abstract. Black-box optimization is a very active area of research, with many new algorithms being developed every year. This variety is needed, on the one hand, since different algorithms are most suitable for different types of optimization problems. But the variety also poses a meta-problem: which algorithm to choose for a given problem at hand? Past research has shown that per-instance algorithm selection based on exploratory landscape analysis (ELA) can be an efficient mean to tackle this meta-problem. Existing approaches, however, require the approximation of problem features based on a significant number of samples, which are typically selected through uniform sampling or Latin Hypercube Designs. The evaluation of these points is costly, and the benefit of an ELA-based algorithm selection over a default algorithm must therefore be significant in order to pay off. One could hope to by-pass the evaluations for the feature approximations by using the samples that a default algorithm would anyway perform, i.e., by using the points of the default algorithm's trajectory. We analyze in this paper how well such an approach can work. Concretely, we test how accurately trajectory-based ELA approaches can predict the final solution quality of the CMA-ES after a fixed budget of function evaluations. We observe that the loss of trajectory-based predictions can be surprisingly small compared to the classical global sampling approach, if the remaining budget for which solution quality shall be predicted is not too large. Feature selection, in contrast, did not show any advantage in our experiments and rather led to worsened prediction accuracy. The inclusion of state variables of CMA-ES only has a moderate effect on the prediction accuracy.

Keywords: Exploratory Landscape Analysis · Automated Algorithm Selection · Black-Box Optimization · Performance Regression · Feature Selection

1 Introduction

In many real-world optimization challenges, we encounter optimization problems which are too complex to be explicitly modeled via mathematical functions, but which nonetheless need to be assessed and solved, more often than not requiring significant computational resources to do so. Explicit problem modeling is also an issue when the relationship between decision variables and solution quality cannot be established other than by simulations or experiments. A standard example for the latter is the design of (deep) neural networks. *Black-box optimization algorithms (BBOA)* are algorithms designed to solve problems of the two types above. BBOA are usually iterative procedures, which actively steer the search by using information obtained from previous iterations, with the goal to eventually converge towards an estimated optimal solution. In each generation, a number of solutions candidates are generated and undergo evaluation.

Classically, BBOA were manually designed, based on users' experience. A plethora of algorithmic components exist from which users can choose to build their own algorithms, and the number of these components is growing every year. Even though the basic underlying principles of these components can be considered similar in nature, their performances on different problem instances can greatly vary. An important and challenging task is thus to select the most appropriate and efficient algorithm when presented with a new, unknown problem instance. This research problem, formalized as the *algorithm selection problem (ASP)* [Ric76], is one of *the* core questions that evolutionary computation aims to answer. The algorithm selection problem is classically tackled by relying on expert knowledge of both the problem instance and the algorithm's strengths and weaknesses. In recent years, however, due to the significant progress in the machine

learning (ML) field, there has been a shift towards an *automated* selection [KKB+18,KT19a,MSKH15,JD20] and configuration [BDSS17] of algorithms based on supervised learning approaches. The idea behind these approaches is in utilizing ML techniques to design and to train models to accurately predict the performance of different black-box algorithms on previously unseen problem instances, with the goal to use these performance predictions to select and to configure the best algorithm for the problem at hand, respectively. In order to apply supervised learning, problem instances need to be represented in a convenient way via numerical values. That is, we need to quantify relevant characteristics of a problem instance through appropriate measures. These measures are referred to as *features*. In the terminology used in evolutionary computation (EC), features are hence aimed at describing the *fitness landscape* of a problem instance.

Fitness landscape analysis has a long tradition in EC. For practical use in black-box optimization, however, the fitness landscape properties can only be described via an informed guessing strategy. Concretely, we can only approximate the fitness landscapes, through the samples that we have evaluated and to which a solution quality has been assigned. Research addressing efficient ways to characterize problem instances via feature approximations is subsumed under the umbrella term *exploratory landscape analysis (ELA)* [MBT+11]. Common research questions in ELA concern the number of samples needed to accurately approximate feature values, the design and the selection of features that are descriptive and easy to approximate, and the possibility to use feature values to transfer learned policies from some instances to previously unseen ones.

A drawback of existing ELA-approaches are the resources needed to extract and to compute the feature values, the time required to train the models, and a lack of explainability. In this work, we focus on the first issues, the feature extraction. Most ELA-based studies perform a three-step selection/configuration: in the first step a number of search points are sampled (commonly using uniform sampling, Latin Hypercube Designs, or quasi-random sampling [RDDD20]), evaluated, and then plugged into a feature computation algorithm such as the R tool *flacco* [KT19b]. In the second step the model for the classification or regression task is built and an algorithm and/or its configuration is suggested. In the third step, this algorithm is then run on the problem instance under consideration. Clearly, the effort for steps 1 and 2 cannot be neglected, and can have a decisive influence on the usefulness of a per-instance algorithm selection/configuration approach, as its effort needs to pay off compared to the performance of a default solver. Even when neglecting the computational overhead of this approach and focusing on function evaluations only as performance measure (as is commonly done in evolutionary computation [HAR+20]), the evaluations needed for completing step 1 need to be taken into account. Much research has been done on determining a suitable number of samples, and typical recommendations vary between $30d$ [BDSS16] and $50d$ samples [KPWT16], where d denotes the dimension of the problem. This is hence a considerable investment.

Of course, one could use these samples to *warm-start* the optimization heuristics, e.g., by initiating them in good regions and/or by calibrating their search behavior based on the information obtained from the samples used to compute the features.

A charming, yet straightforward alternative would be to integrate the first step of the ELA-based approach described above into the optimization routine, by computing the features based on the search points that a default algorithm would anyway perform. That is, one would use the search trajectory of such a default algorithm to predict and then to select and/or to configure a solver *on the fly*, once or even several times during the optimization process.

Similar to parameter control [KHE15,AM16,DD20], such a dynamic selection would not only allow to *identify* an efficient algorithm for the given problem instance, but could also benefit from *tracking* the best choice while the optimization process (and the best response to its needs) evolves. Such a dynamic algorithm selection can therefore be seen as an ELA-based variant of hyper-heuristics [BGH+13]. The approach has previously been used in the context of constrained optimization, with the goal to have a dynamic, ELA-guided selection of a suitable constraint handling technique; see [Mal18] for examples and further references.

A key challenge in applying dynamic ELA-based algorithm selection is the fact that the feature values can vary drastically between different sampling strategies [RDDD20]. Since the distribution of points sampled on different problem instances can differ quite drastically even when using the same algorithm, it is not clear, a priori, if or how suitable ML models can be trained. This challenge was confirmed in [JD19], where it was shown that the landscape which an algorithm sees locally during the optimization process (i.e., the partial

landscape it is aware of at each step of the optimization process) usually differs a lot from the global fitness landscape of the problem it is solving.

Our Results. With the long-term goal to obtain well-performing dynamic ELA-based algorithm selection and configuration techniques, we analyze in this work a first, rather cautious task: ELA-based performance prediction using the trajectory samples of the algorithm under investigation. More precisely, we consider the Covariance Matrix Adaptation Evolution Strategy (CMA-ES [HO01]), and we aim at predicting its solution quality (measured as target precision, i.e., the difference to an optimal solution in quality space) after a fixed budget of function evaluations. Concretely, we use the first 250 samples evaluated by the CMA-ES and we aim at predicting its performance after additional 250 evaluations, doing so for 20 independent CMA-ES runs. The performance regression is done via a random forest model which takes as input the features computed from the trajectory data and which outputs an estimate for the final solution quality.

We then take into account that problem characteristics cannot only be described via classic ELA features, but that internal states of the search heuristics can also be used to derive information about the problem instance at hand. Such approaches have in the past been used, for example, for local surrogate-modelling [PRH19]. We analyze the accuracy gains when using the same state information as in [PRH19], that is, the values of the CMA-ES internal variables that mainly carry information about the current probability distribution from which the CMA-ES samples candidates for the new generation. In our experiments, the advantage of using this state information over using ELA-features only, however, is only marginal. Concretely, the average difference between true and predicted solution quality decreases from 14.4 to 12.1 when adding the state variables as features (where the average error reported here is taken over all 24 benchmark problems from the BBOB suite of the COCO platform [HAR$^+$20], and over all performed CMA-ES runs).

We observe in the experiments above that some CMA-ES runs are drastic outliers in terms of performance, at times with the target precision differing from the target precision of all the other runs by up to 10 orders of magnitude. We therefore also consider an intentionally more "friendly" setting, in which we analyze the regression quality only for the run achieving median performance on a given problem instance. Conclusions for combining trajectory-based and state variable features remain almost identical to those stated above.

We then compare these median trajectory-based predictions to the classical approach using globally sampled features. Here, we pessimistically assume that the samples were computed for free. That is, we couple 2 separate sets of the global feature values approximated from 250 and 2000 uniformly sampled points each to the target precision achieved by the CMA-ES after 500 function evaluations. Interestingly, the difference in prediction accuracy compared to our trajectory-based predictions is rather small. The global predictions still remain, however, more accurate, with an average absolute prediction error of 4.7 vs. 6.2 for the trajectory approach (where again the average is taken over all 24 BBOB functions).

Furthermore, we also use this median setting to analyze the influence of feature selection on prediction accuracy. Different state-of-the-art methods were applied, using a transfer learning scenario, to select features estimated to be the most important and to have highest discriminative power. Here again, the differences in prediction accuracy were small, with feature selection surprisingly leading to an overall slightly worsened solution quality than the full feature portfolio.

As suggested in [JD20], all our experiments are based on two independently trained models: one which aims to predict target precision after 500 evaluations, and one which predicts the logarithm of this target precision. While the former is better in guessing the broader "ball park", the latter is more suitable for fine-grained performance prediction, i.e., when the expected performance of the algorithm is very good. As in [JD20], we also build a combined regression, which uses either one of the two models, depending on whether the predicted performance is better or worse than a certain threshold. The optimal thresholds differ quite drastically between different feature sets. However, a sensitivity analysis reveals that their influence on overall performance is rather small. Also, the ranking of the different feature portfolios remains almost unaffected by the choice of the threshold. In line with the results in [JD20], the combined models perform consistently better than any of the two standalone ones, albeit slightly.

2 Supervised ML for Performance Regression

The Experimental Setup. When it comes to landscape-aware performance prediction, supervised machine learning techniques such as regression and classification have been studied in a variety of settings. Regression models, unlike classification ones, have an advantage of keeping track of the magnitude of differences between performances of different algorithms, as they measure concrete values for performances of all algorithms from the portfolio.

Among different supervised learning regressors in the literature, such as support vector machines, Gaussian processes or ridge regression to name a few, it has been empirically shown that random forests outperform other models in terms of prediction accuracy [HXHL14]. A random forest is an ensemble-based meta-estimator that works by fitting multiple decision trees on subsamples of the original data set, then uses averaging as a way to control overfitting. In our experimental setup, we used an off-the-shelf random forest regressor from the Python *scikit-learn* package [PVG+11], without parameter tuning and using 1000 estimators.

We restricted this work to a single heuristic, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES [HO01]). The CMA-ES works by iteratively sampling a new population of candidate solutions from a shifted multivariate normal distribution, choosing the best offspring of the current population based on their respective fitness values, and then updating the parameters of the probability distribution according to the best candidates. For the purpose of our work, we used its standard version, available in the Python *pycma* package [HAB19], which uses a fixed population size and no restarts during the optimization process.

As our benchmark, we used the first five instances of all 24 noiseless *BBOB* functions of the *COCO* platform [HAR+20], an environment for comparison of algorithm performance in continuous black-box optimization. The different instances of each function are generated by translating and rotating the function in the objective space. These transformations do not affect the performance of CMA-ES, but they do influence some of the feature values, especially those which are not transformation-invariant [SEK20] (for the invariant features, the boundary handling can have an effect on the feature values). We focus on dimension $d = 5$ here.

For our first experiments, we perform 20 independent runs of the CMA-ES on these 120 problem instances, while keeping track of the search trajectories and the internal state variables of the algorithm itself. Throughout this work, we fix the budget of 500 function evaluations, after which we stop the optimization and record the target precision of the best found solution within the budget. In order to predict those recorded target precisions after 500 function evaluations, we compute the trajectory-based landscape features using the first 250 sampled points and their evaluations from the beginning of each trajectory, and couple them with the values of the internal CMA-ES state variables extracted at the 250^{th} function evaluation.

Figure 1 summarizes the target precision achieved by CMA-ES in each of the 20 runs. We see that the results are more or less homogeneous across different runs and across different instances of the same problem. However, we also observe several outliers, e.g., for functions 7 (outlier for all instances), function 10 (instance 4), function 12 (instance 1). It is important to keep in mind that the randomness of these performances are entirely caused by the randomness of the algorithm itself – the problem instance does not change between different runs.

For landscape feature computation, we use the R package *flacco* [KT19b]. Following suggestions made in [KT19a,BDSS17] we restrict ourselves to those feature sets that do not require additional function evaluations for computing the features. Namely, in this work we use 2 original ELA feature sets (*y-Distribution* and *Meta-Model*), as well as *Dispersion*, *Nearest-Better Clustering* and *Information Content* feature sets. This gives us a total of 38 landscape features per problem instance. In addition, we follow up on an idea previously used in [PRH19] and consider a set of internal CMA-ES state variables as features:

- Step-size: its value indicates how large is the approximated region from which the CMA-ES samples new candidate solutions.
- Mahalanobis mean distance: represents the measure of suitability of the current sampled population for model training from the point of view of the current state of the CMA-ES algorithm.
- C evolution path length: indicates the similarity of landscapes among previous generations.
- σ evolution path ratio: provides information about the changes in the probability distribution used to sample new candidates.

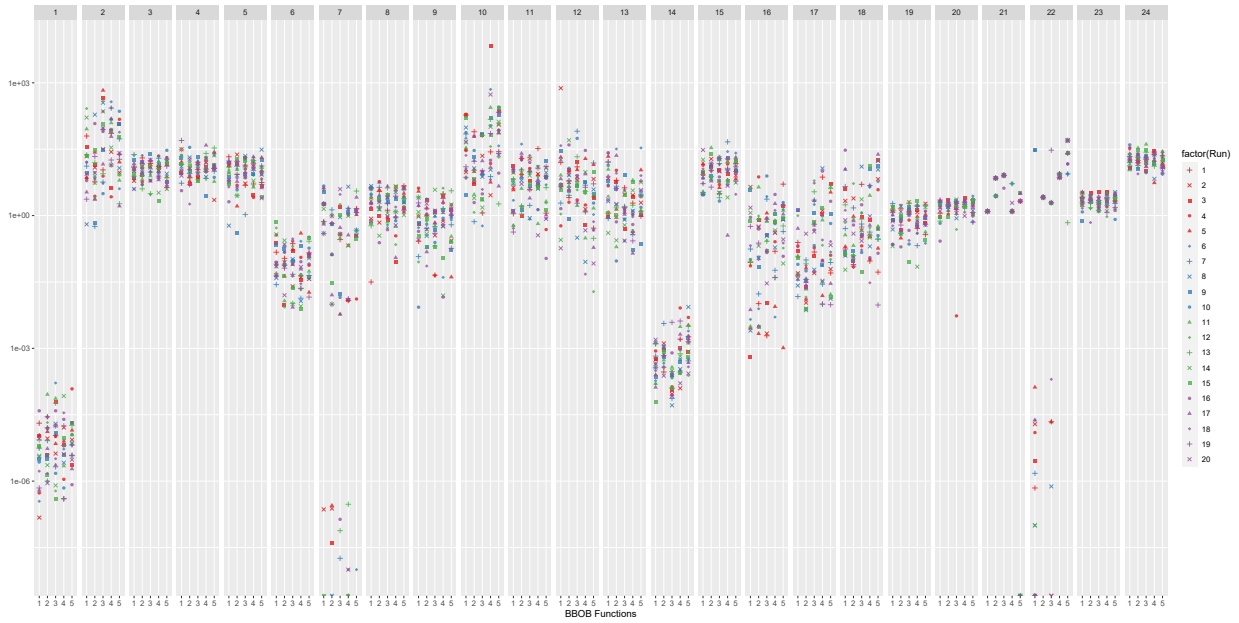

Fig. 1. Target precision achieved by the CMA-ES with a budget of 500 function evaluations, for each of the first five instances of all 24 BBOB functions. Differently colored and shaped points represent 20 independent CMA-ES runs.

– CMA similarity likelihood: it is a log-likelihood of the set of candidate solutions with respect to the CMA-ES distribution and may also represent a measure of the set suitability for training.

As suggested in [JD20], and using the elements described above, we establish two separate regression approaches. One model is trained to predict the actual, true value of the target precision data (we refer to it as the *unscaled model* in the remainder of the paper), while the other predicts the logarithm of the target precision data (the *log-model*). It is important to note that the target precision measure intuitively carries the information about the order of magnitude of the actual distance to the optimum, i.e., the *distance level* to the optimum, which is effectively computed as the log-target precision. For instance, if an algorithm reaches a target precision of 10^{-3} for one problem instance and 10^{-7} for another, it means that the algorithm found a solution which is 4 distance levels closer to the optimum in the latter scenario. Moreover, to reduce variability, we estimate both models' prediction accuracy through performing a 5-fold *leave-one-instance-out* cross-validation, making sure to train on 4 out of 5 instances per BBOB function, test on the remaining instance and combine the results over the rounds.

Results. Adopting our two regression models, we trained them separately in the following three scenarios: using as predictor variables the landscape features only, using the internal CMA-ES state variables only, and using the combination of the two. We trained the random forests 3 independent times and took a median of the 3 runs to ensure the robustness of the results.

Figure 2 highlights the absolute prediction errors per BBOB function using two regression models, the unscaled and the log- one, when trained with 3 different feature sets: using only the trajectory landscape data, only the CMA-ES state variable data, and the combination of the two. For the majority of the functions, using the combination of the trajectory data and the state variable data seems to help in improving the performance prediction accuracy, compared to the scenarios which use only one of those two feature sets.

We also confirm that the log-model is indeed better at predicted fine-grained target precision (e.g., in the case of F1 (sphere function) or F6 (linear slope function), we know that those functions do not require many function evaluations to converge to the global optimum, and their recorded target precision values are already quite small as they are very near the optimal solution). On the other hand, the unscaled model performs better where the target precision values are higher (e.g., for the functions such as F3, F15 (two

	Unscaled model			Log model		
FID	SV	ELA	ELA+SV	SV	ELA	ELA+SV
1	4.6	2.6	1.9	0.1	0.0	0.0
2	60.4	64.2	63.7	65.4	58.0	57.1
3	6.4	5.4	5.2	10.1	7.2	6.9
4	9.0	7.0	6.9	12.4	8.7	8.6
5	86.4	6.6	26.2	5.5	4.6	4.5
6	9.1	9.4	4.0	1.1	0.5	0.6
7	6.5	1.0	0.8	0.9	0.7	0.7
8	6.8	1.8	1.7	1.7	1.2	1.2
9	9.2	1.1	1.0	0.9	0.7	0.7
10	126.2	171.1	125.1	125.9	119.7	114.8
11	37.2	12.7	23.9	6.8	5.4	5.6
12	15.2	101.8	24.5	15.3	14.3	13.0
13	9.8	5.6	5.1	4.1	3.9	3.7
14	4.8	2.5	1.1	0.4	0.1	0.0
15	8.8	10.7	5.5	9.6	6.9	6.5
16	4.7	4.8	5.0	0.8	1.7	1.7
17	6.4	1.9	1.8	1.4	0.7	0.7
18	7.8	2.4	2.3	2.7	2.3	2.3
19	11.7	2.3	1.6	2.9	0.5	0.5
20	4.6	0.5	0.6	1.4	0.6	0.6
21	3.8	3.5	3.7	3.5	3.6	3.6
22	10.1	9.0	8.6	9.1	8.6	8.6
23	13.4	1.8	2.0	2.6	0.8	0.8
24	9.4	5.2	5.1	14.9	8.3	8.5

Fig. 2. Absolute prediction errors for both regression models aggregated per BBOB function in 3 different scenarios depending on the feature set used. The *SV* column stands for the CMA-ES state variables, the *ELA* for the landscape features, and the third one is the combination of both.

versions of Rastrigin function), and also F24 (Lunacek bi-Rastigin), which are all highly multimodal, the number of function evaluations in our budget was not nearly enough to allow for finding a true optimum).

We also notice that using only the state variables for the unscaled model does not suffice for an accurate prediction in the most cases. The reverse situation is nevertheless also possible: we see that for F12, using only the state variables yields the best accuracy in the unscaled model. Furthermore, there are also exceptions where using only the landscape data results in a higher accuracy than using the combined features (e.g., F11 for both models, F5 for the unscaled model).

3 Comparison with Global Feature Values

We then proceeded to compare the differences in the prediction accuracy from the sets described in the Section 2 with the prediction accuracy using the global feature data, both alone and combined with the same CMA-ES state variable data as above. To be able to perform a fair comparison, for the trajectory data we selected from the 20 executed CMA-ES runs those runs with the median target precision value per problem instance and their corresponding features and re-trained the unscaled and the log-model. Global features-wise, both models were also trained using features computed from 2000 and 250 globally uniformly sampled points (the median value of 50 independent feature computations) for each function and instance.

Figure 3 shows the absolute errors in prediction when the trajectory-based approach is compared with the results using the global features. The highest accuracy is reported in cases when only the global landscape features were used, across almost all problems, with 2000-sample features yielding the best results. Here, we do not observe a huge improvement when combining the global landscape features with the state variable data. It seems that the number of samples used to compute the features can be crucial in reducing the

	Unscaled model							Log model						
FID	SV	ELA	ELA +SV	GLOB2k	GLOB2k +SV	GLOB250	GLOB250 +SV	SV	ELA	ELA +SV	GLOB2k	GLOB2k +SV	GLOB250	GLOB250 +SV
1	4.4	2.2	2.0	0.9	1.3	1.8	2.0	0.29	0.00	0.00	0.00	0.00	0.00	0.00
2	31.2	17.4	16.7	18.5	17.9	18.9	17.7	38.89	31.56	31.86	25.65	26.58	28.98	30.20
3	8.3	5.2	5.5	1.6	2.6	3.6	4.9	10.14	7.08	6.97	4.93	5.69	8.33	8.80
4	9.3	3.5	3.8	2.9	3.7	3.9	4.6	11.78	4.45	4.99	3.59	4.87	7.64	8.31
5	1.7	1.4	1.0	1.3	1.1	1.2	0.9	7.16	8.19	8.17	4.97	5.22	5.93	6.21
6	5.0	6.7	6.1	1.4	1.6	2.5	2.5	0.66	1.45	1.33	0.23	0.25	0.30	0.34
7	3.8	4.9	4.7	13.1	12.9	4.0	3.2	0.68	0.31	0.33	2.42	2.05	1.00	1.01
8	3.2	1.6	1.9	1.9	2.2	1.8	2.0	0.98	0.98	0.96	0.81	0.76	0.91	0.71
9	3.2	1.1	0.7	0.5	0.4	1.0	0.8	0.73	0.26	0.27	0.45	0.38	0.40	0.41
10	37.0	24.4	24.3	19.5	19.3	24.8	26.9	37.98	30.84	31.48	29.15	29.94	28.75	30.17
11	14.4	8.6	10.2	2.6	3.0	6.4	8.4	2.18	5.18	5.36	3.09	3.52	2.57	3.05
12	2.2	24.9	20.2	2.3	2.5	4.4	3.7	4.55	1.76	1.40	3.43	3.57	3.65	3.55
13	9.7	2.2	3.6	2.4	2.4	3.2	4.8	1.95	1.62	1.70	1.73	1.81	1.49	1.65
14	2.8	0.7	0.6	3.5	3.5	5.5	5.4	0.45	0.00	0.00	0.05	0.06	0.23	0.25
15	5.5	3.6	3.5	4.3	5.5	5.6	5.8	7.23	7.48	6.23	6.04	6.57	8.23	8.40
16	2.8	5.6	5.5	0.6	1.1	0.5	1.1	0.35	4.26	4.08	0.27	0.33	0.21	0.23
17	3.3	1.3	1.2	2.4	2.2	3.3	2.8	0.89	0.26	0.25	0.27	0.30	0.27	0.27
18	3.3	0.9	1.0	3.0	2.7	3.6	3.0	0.44	0.34	0.27	0.29	0.25	0.39	0.41
19	18.1	2.7	3.5	1.4	2.3	1.4	3.2	2.91	0.24	0.27	0.43	0.53	0.40	0.43
20	3.3	5.8	5.7	0.2	1.1	0.6	1.9	1.27	0.67	0.80	0.67	0.93	0.64	0.75
21	3.1	1.9	2.6	4.1	3.6	2.9	3.0	3.27	3.16	3.15	3.96	3.98	4.11	3.96
22	7.2	7.0	7.0	9.1	9.2	9.6	9.3	7.87	7.45	7.45	7.66	7.85	7.61	7.66
23	6.9	3.0	3.6	0.2	0.9	0.5	2.5	2.53	1.11	1.19	0.61	0.63	0.65	0.68
24	6.7	4.0	4.2	1.5	2.4	2.2	3.3	12.65	8.35	9.22	4.41	7.24	8.07	9.87

Fig. 3. Absolute prediction errors for both regression models for the median trajectory-based prediction (the first 3 columns of each block) and the median global feature prediction (the middle two columns of each block represent the errors when using the 2000-sample features, and the last two columns correspond to using the 250-sample features).

errors in prediction, as global sampling could be linked to a potential higher discriminative power of features thus computed. Again, for certain functions such as F2 and F10 (both of which are different variants of the ellipsoidal function), we observe an overall low accuracy.

4 Sensitivity Analyses

Feature Selection. To provide a sensitivity analysis based on the features used for the performance regression, we performed feature selection in the scenario of transfer learning, i.e., between different supervised tasks, where the features selected for the problem classification task have been evaluated on the performance regression task.

To do this, we have explored four state-of-the-art feature selection techniques: *Boruta* [KJR10] is a feature selection and ranking algorithm based on random forests algorithm, which only selects features that are statistically significant. *Recursive feature elimination (rfe)* [GFBG06] learns a model assessing different sets of features by recursively eliminating features per loop until a good model is learnt. It requires an ML algorithm for evaluation, and here we use a random forest. *Stepwise forward and backward selection (swfb)* [DK92] tries to fit the best regression model by iteratively selecting and removing features. In our experiments, we used it in both directions simultaneously. *Correlation analysis with different threshold values (cor)* [BCHC09] is based on the correlation analysis done only using the features (i.e., excluding the target). The result is a feature set where highly correlated features are omitted. In our case, we tested three different correlation thresholds: 0.50, 0.75, and 0.90. Note that while the first three feature selection methods require a supervised ML task, the last one is completely unsupervised and does not depend on the target.

	boruta	swfb	rfe	cor0.5	cor0.75	cor0.9
# selected ELA features	37	1	7	4	9	15
# selected state variable features	2	0	0	3	3	5

Table 1. Number of ELA and state variable features for each selected feature portfolio. Details are available in Table 3.

			SV	**ELA**	**ELA +SV**	**GLOB2k**	**GLOB2k +SV**	**GLOB250**	**GLOB250 +SV**	**boruta**	**cor 0.5**	**cor 0.75**	**cor 0.9**	**rfe**	**swfb**
								Best threshold τ							
FID	**min_tp**	**max_tp**	**1.336**	**3.99**	**4.742**	**14.497**	**9.46**	**0.694**	**2.605**	**3.63**	**1.813**	**4.901**	**1.717**	**7.388**	**20**
1	0	0	0.43	0	0	0	0	0	0	0	0	0	0	0	0.21
2	9.25	87.47	44.69	25.47	24.46	24.49	23.65	28.96	27.98	25.16	36.78	35.28	34.62	24.51	53.85
3	10.34	14.63	9.73	6.62	7.02	5.62	6.25	4.43	9.39	6.73	8.08	7.7	5.63	7.95	8.82
4	10.29	14.53	11.86	5.08	5.08	4.81	5.73	5.1	7.49	5.18	9.26	6.71	5.59	7.53	8.41
5	8.53	11.34	4.59	8.24	8.15	6.02	5.52	1.48	1.06	8.21	2.36	6.76	4.91	4.52	7.74
6	0.04	0.11	0.78	3.87	2.15	0.24	0.27	0.33	0.37	4.46	1.02	1.19	1.65	0.89	1.43
7	0.13	1.83	3.84	0.36	0.39	4.49	4.07	4.62	1.13	0.42	0.86	0.6	0.65	0.67	7.94
8	1.22	2.59	3.02	1.06	1.1	1.25	1.22	2.03	0.84	0.95	2.02	0.93	1.04	1.42	4.88
9	0.68	1.36	3.05	0.34	0.34	0.59	0.53	0.77	0.48	0.33	0.52	0.26	0.29	0.86	3.47
10	8.44	84.69	43.33	32.06	32.85	21.39	23.16	29.29	33.35	32.13	42.18	32.73	35.01	29.64	47.68
11	4.97	8.83	17.48	10.37	12.63	3.46	2.99	8.26	10.18	11.17	25.11	21.61	20.86	2.43	4.77
12	2.53	6.32	4.76	19.34	15.89	4.03	4.16	5.76	4.13	19.45	4.38	2.31	9.69	19.32	3.56
13	1.09	5.92	13.62	2.07	2.13	2.29	2.32	3.51	2.21	2.11	2.45	2.04	5.37	2.03	3.5
14	0	0	1.65	0.01	0.01	0.07	0.08	1.89	0.37	0.01	0.61	0	0	0	5.18
15	8.49	12.93	7.44	6.79	6.66	6.08	6.81	6.24	8.12	5.77	7.07	6.11	5.88	6.73	9.12
16	0.18	0.83	0.87	6.31	5.85	0.32	0.35	0.25	0.25	6.45	0.48	4.42	4.77	2.43	1.21
17	0.03	0.64	2.63	0.36	0.33	0.32	0.33	0.28	0.31	0.38	4.25	0.41	0.38	0.27	2.75
18	0.16	0.69	1.54	0.34	0.34	0.33	0.3	4.96	0.45	0.34	0.54	0.28	0.3	0.54	1.24
19	0.75	1.19	18.75	0.34	0.36	0.65	0.67	1.66	0.49	0.38	9.95	0.59	0.5	0.46	8.54
20	1.7	1.79	3.77	0.76	0.84	0.99	1.17	0.99	0.93	0.69	3.2	1.22	3.16	1.77	5.16
21	0	8.12	4.86	4.55	4.52	5.02	5.02	5.2	4.95	4.53	4.72	4.84	4.83	4.83	3.72
22	0	25.48	12.04	11.91	11.91	11.9	11.91	12.88	11.94	11.91	11.89	11.93	11.7	11.92	5.34
23	1.99	2.35	7.49	3.67	3.61	0.69	0.77	0.64	0.91	3.72	7.89	2.73	4.53	2.46	3.04
24	15.73	20.73	10.06	4.76	4.94	5.15	6.52	2.95	4.29	4.81	11.13	9.49	7.22	7.43	11.65
Overall RMSE, combined			**15.05**	**10.41**	**10.25**	**7.74**	**7.92**	**9.48**	**10.05**	**10.43**	**13.66**	**11.67**	**11.86**	**9.77**	**15.73**
Overall RMSE, unscaled			15.08	11.18	10.88	9.21	9.30	9.58	10.19	11.16	14.11	11.80	12.00	10.93	17.03
Overall RMSE, log			15.63	13.05	13.21	11.46	11.88	12.05	12.87	13.14	14.65	13.89	14.29	12.61	15.73

Table 2. RMSE values of the combined selector in three scenarios: when the prediction is based on the search trajectory landscape features and state variables (first 3 columns), on global features (next 4 columns), and finally on selected feature portfolios (last 6 columns).

Our experimental design has been done using stratified 5-fold cross-validation. For a fair feature selection, we used the aforementioned methods on each training fold separately, then selected the intersection of the features returned by each training fold in the end. These features are further evaluated in the performance regression task.

Table 1 summarizes how many features were selected per portfolio, from the whole set of 38 ELA landscape features and 5 CMA-ES state variable features.

Combined Selector Model and Sensitivity Analysis.

As common in ML, we measure the regression accuracy in terms of *Root Mean Squared Error* (RMSE). Table 2 summarizes the RMSE values for the different feature portfolios when using (1) the unscaled model, (2) the scaled model, and (3) a combination of unscaled and the log-model (see last three rows of Table 2). The threshold τ at which the predictive model changes is optimized for each feature portfolio individually, the obtained thresholds are summarized at the top of Table 2. That is, we select the prediction of the log-model when the predicted precision (according to the log-model) is smaller than the threshold value τ, and we use the prediction of the unscaled model otherwise. Note that the optimal threshold value τ varies significantly between the different feature portfolios.

When comparing all the different portfolios (initial trajectory-based, global and selected trajectory-based ones), the good performance of the global feature sets is not surprising. Differences from the initial trajectory-

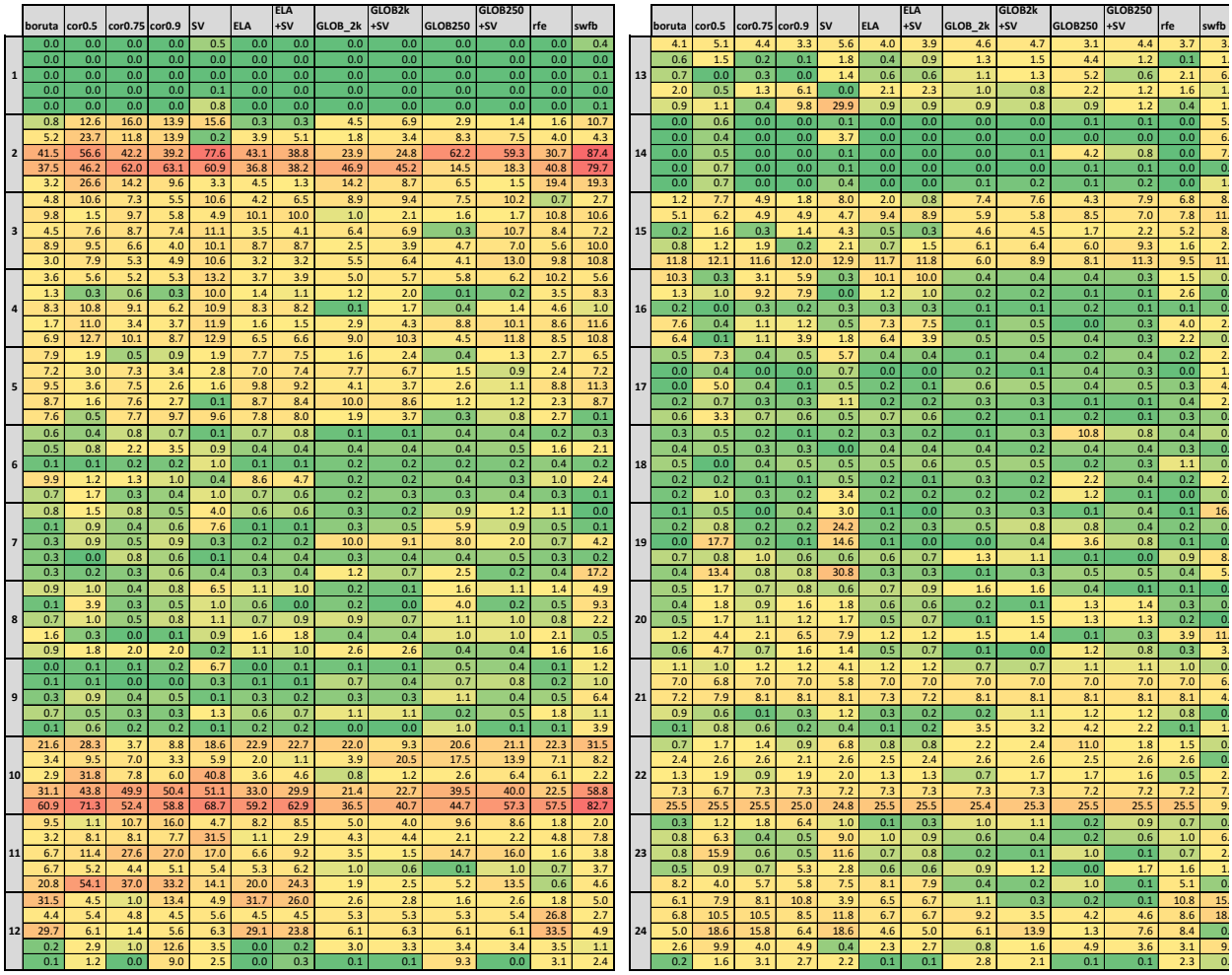

Fig. 4. Absolute prediction errors of the combined models using portfolio-specific optimal thresholds τ.

based predictions are marginal for sets such as *boruta*, *cor0.75* and *cor0.9*, whereas *swbf* and *cor0.5* perform constantly worse than *ELA+SV*. Using the *rfe* set, on the other hand, led to better results than using the original feature set. *SV* alone does not achieve good accuracy, but its contribution to ELA-only feature portfolio is around 3% at the best threshold for the combined model, which is $\tau = 4.901$. The absolute errors per instance are plotted in Figure 4.

5 Conclusions and Future Work

We analyzed in this paper the accuracy of predicting the CMA-ES solution quality after given budget based on the features computed from the samples on the CMA-ES search trajectory using two complementary regression models, the unscaled and the log-model. Adding information obtained from the CMA-ES internal state variables does not improve the prediction accuracy drastically compared to the trajectory-based data only. Those results were then contrasted to the regression using the global features, where using the latter ones, especially those computed using a higher number of samples, yielded a consistently better accuracy.

Next, we tested whether we would achieve further gains in accuracy through feature selection. Although the overall results are comparable to the ones from initial trajectory-based portfolios, several selected feature

sets resulted in worse accuracy than in the initial approach. We ultimately pointed out the advantages of using our combined selector model over relying separately on predictions of the standalone unscaled or log-model across all different feature portfolios in all 3 scenarios.

In terms of **future work,** we plan on continuing this research by considering the following questions and tasks:
(0) Performance prediction of **other solvers**: How accurately can we use trajectory-based features of one algorithm to predict the performance of another algorithm? In this work, we have only tried to predict performance for the same algorithm from whose trajectory the feature values have been computed. A next step would be to test if models for configuring the same algorithm can be trained. When this is successful, transfer learning from one algorithm to another one can be considered.
(1) How can we more efficiently capture the **temporal component**, i.e., the information which sample was evaluated *when* during the search? Using such longitudinal data, both in terms of extracted feature values and in terms of state variable evolution could possibly be done using recurrent neural networks [ZFW$^+$19].
(2) **Combining global and trajectory-based sampling:** In our work, we only considered the case in which *either* global sampling *or* trajectory-based sampling is used. The accuracy of the models based on global sampling was better than that of the trajectory-based features. Even if we keep in mind that this comparison was unfair in that we provided the global feature values "for free", the results nevertheless suggest that a combination of global and trajectory-based feature computations could be worthwhile to investigate. How we can optimally balance the budget between global sampling, trajectory-based sampling, and remaining optimization budget is a challenging question in this context.
(3) **Warm-starting** the CMA-ES such that it starts the optimization process with the covariance matrix and other parameters that are extrapolated from the (uniformly or otherwise) distributed global samples might significantly improve the overall accuracy, as the CMA-ES will have a better overview of the whole problem instance "from the get-go". A similar approach has been suggested in [MRT15] when switching from a Bayesian optimization algorithm to CMA-ES.
(4) **Feature selection and ranking:** Instead of using transfer learning for feature selection between two different supervised ML tasks, feature selection within the same supervised task has not been considered in this paper. We also plan on making better use of variable importance estimations provided by feature ranking algorithms such as those based on ensemble of predictive clustering trees [PKD20] and those based on ReliefF and RReliefF [RŠK03].
(5) **Feature design:** The work [DLV$^+$19] suggests several algorithm-specific features for the SOO tree algorithm [Mun11]. Such specific features can much more explicitly capture the characteristics of the algorithm-problem instance interaction. It could be worthwhile to study whether, possibly in addition to the longitudinal data mentioned in (1), such specific features can be identified for other common solvers, such as the CMA-ES.
(6) **Feature portfolio:** We note that our work above is based on the features available in the *flacco* package [KT19b]. Since the design of *flacco*, however, several new feature sets have been suggested. Another straightforward way to extend our analyses would be in the inclusion of these feature sets, with the hope to improve the overall regression accuracy. In this respect, we find in particular the Search Trajectory Networks suggested in [OMB20] worth investigating.
(7) **Representation learning of landscapes:** The feature data will be additionally explored by applying representation learning methods that automatically learn new data representations by reducing the dimension of the data, automatically detecting correlations, and removing bias and redundancies presented in the feature data. The work presented in [EPR$^+$20] showed that linear matrix factorization representations of the ELA features values significantly detects better correlation between different problem instances.
(8) **Hyperparameter tuning of regression models:** Last, but not least, we are planning to explore algorithm portfolio that consists of different regression methods in order to find the most suitable one, together with finding its best hyperparameters for achieving better performance. In this study, we have used random forest for regression without tuning its parameters, since we have been interested in the contribution of different feature portfolios.

Acknowledgments. This research benefited from the support of the Paris Ile-de-France region and of a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx

LMH. This work was also supported by projects from the Slovenian Research Agency: research core funding No. P2-0098 and project No. Z2-1867. We also acknowledge support by COST Action CA15140 "Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO)".

References

AM16. Aldeida Aleti and Irene Moser, *A systematic literature review of adaptive parameter control methods for evolutionary algorithms*, ACM Comput. Surv. **49** (2016), 56:1–56:35.

BCHC09. Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen, *Pearson correlation coefficient*, Noise reduction in speech processing, Springer, 2009, pp. 1–4.

BDSS16. N. Belkhir, J. Dréo, P. Savéant, and M. Schoenauer, *Surrogate assisted feature computation for continuous problems*, LION, Springer, 2016, pp. 17–31.

BDSS17. ———, *Per instance algorithm configuration of CMA-ES with limited budget*, GECCO, ACM, 2017, pp. 681–688.

BGH+13. Edmund K. Burke, Michel Gendreau, Matthew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu, *Hyper-heuristics: a survey of the state of the art*, Journal of the Operational Research Society **64** (2013), 1695–1724.

DD20. Benjamin Doerr and Carola Doerr, *Theory of parameter control mechanisms for discrete black-box optimization: Provable performance gains through dynamic parameter choices*, Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, Springer, 2020, pp. 271–321.

DK92. Shelley Derksen and Harvey J Keselman, *Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables*, British Journal of Mathematical and Statistical Psychology **45** (1992), no. 2, 265–282.

DLV+19. B. Derbel, A. Liefooghe, S. Vérel, H. Aguirre, and K. Tanaka, *New features for continuous exploratory landscape analysis based on the SOO tree*, FOGA, ACM, 2019, pp. 72–86.

EPR+20. Tome Eftimov, Gorjan Popovski, Quentin Renau, Peter Korosec, and Carola Doerr, *Linear matrix factorization embeddings for single-objective optimization landscapes*, SSCI, IEEE, 2020, pp. 775–782.

GFBG06. Pablo M Granitto, Cesare Furlanello, Franco Biasioli, and Flavia Gasperi, *Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products*, Chemometrics and Intelligent Laboratory Systems **83** (2006), no. 2, 83–90.

HAB19. Nikolaus Hansen, Youhei Akimoto, and Petr Baudis, *CMA-ES/pycma on Github*, https://github.com/CMA-ES/pycma, 2019.

HAR+20. Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff, *COCO: a platform for comparing continuous optimizers in a black-box setting*, Optimization Methods and Software (2020), 1–31.

HO01. Nikolaus Hansen and Andreas Ostermeier, *Completely derandomized self-adaptation in evolution strategies*, Evolutionary Computation **9** (2001), no. 2, 159–195.

HXHL14. Frank Hutter, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown, *Algorithm runtime prediction: Methods & evaluation*, Artif. Intell. **206** (2014), 79–111.

JD19. Anja Jankovic and Carola Doerr, *Adaptive landscape analysis*, GECCO, Companion Material, ACM, 2019, pp. 2032–2035.

JD20. ———, *Landscape-aware fixed-budget performance regression and algorithm selection for modular CMA-ES variants*, GECCO, 2020, pp. 841–849.

KHE15. Giorgos Karafotias, Mark Hoogendoorn, and A.E. Eiben, *Parameter control in evolutionary algorithms: Trends and challenges*, IEEE Trans. Evol. Comput. **19** (2015), 167–187.

KJR10. Miron B Kursa, Aleksander Jankowski, and Witold R Rudnicki, *Boruta–a system for feature selection*, Fundamenta Informaticae **101** (2010), no. 4, 271–285.

KKB+18. P. Kerschke, L. Kotthoff, J. Bossek, H.H. Hoos, and H. Trautmann, *Leveraging TSP solver complementarity through machine learning*, Evolutionary Computation **26** (2018), no. 4.

KPWT16. P. Kerschke, M. Preuss, S. Wessing, and H. Trautmann, *Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models*, GECCO, 2016, pp. 229–236.

KT19a. P. Kerschke and H. Trautmann, *Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning*, Evolutionary Computation **27** (2019), no. 1, 99–127.

KT19b. Pascal Kerschke and Heike Trautmann, *Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco*, Applications in Statistical Computing – From Music Data Analysis to Industrial Quality Improvement (Nadja Bauer, Katja Ickstadt, Karsten Lübke, Gero Szepannek, Heike Trautmann, and Maurizio Vichi, eds.), Springer, 2019, pp. 93 – 123.

Mal18. Katherine Mary Malan, *Landscape-aware constraint handling applied to differential evolution*, TPNC, LNCS, vol. 11324, Springer, 2018, pp. 176–187.

MBT+11. O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, *Exploratory Landscape Analysis*, GECCO, ACM, 2011, pp. 829–836.

MRT15. Hossein Mohammadi, Rodolphe Le Riche, and Eric Touboul, *Making EGO and CMA-ES complementary for global optimization*, Proc. LION, Springer, 2015, pp. 287–292.

MSKH15. Mario A. Muñoz, Yuan Sun, Michael Kirley, and Saman K. Halgamuge, *Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges*, Inf. Sci. **317** (2015), 224–245.

Mun11. Rémi Munos, *Optimistic optimization of a deterministic function without the knowledge of its smoothness*, Advances in Neural Information Processing Systems, 2011, pp. 783–791.

OMB20. Gabriela Ochoa, Katherine Mary Malan, and Christian Blum, *Search trajectory networks of population-based algorithms in continuous spaces*, EvoApplications, Springer, 2020, pp. 70–85.

PKD20. Matej Petković, Dragi Kocev, and Sašo Džeroski, *Feature ranking for multi-target regression*, Machine Learning **109** (2020), no. 6, 1179–1204.

PRH19. Zbynek Pitra, Jakub Repický, and Martin Holena, *Landscape analysis of Gaussian process surrogates for the covariance matrix adaptation evolution strategy*, GECCO, ACM, 2019, pp. 691–699.

PVG+11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, JMLR **12** (2011), 2825–2830.

RDDD20. Quentin Renau, Carola Doerr, Johann Dréo, and Benjamin Doerr, *Exploratory landscape analysis is strongly sensitive to the sampling strategy*, PPSN, Springer, 2020, pp. 139–153.

Ric76. John R. Rice, *The algorithm selection problem*, Advances in Computers, vol. 15, Elsevier, 1976, pp. 65 – 118.

RŠK03. Marko Robnik-Šikonja and Igor Kononenko, *Theoretical and empirical analysis of relieff and rrelieff*, Machine learning **53** (2003), no. 1-2, 23–69.

SEK20. U. Skvorc, T. Eftimov, and P. Korosec, *Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis*, Appl. Soft Comput. **90** (2020), 106138.

ZFW+19. Juan Zhao, QiPing Feng, Patrick Wu, Roxana A Lupu, Russell A Wilke, Quinn S Wells, Joshua C Denny, and Wei-Qi Wei, *Learning from longitudinal data in electronic health record and genetic data to improve cardiovascular event prediction*, Scientific reports **9** (2019), no. 1, 1–10.

Feature	# sets	boruta	swfb	rfe	cor0.5	cor0.75	cor0.9
ela_distr.skewness	4	x	x	x			x
ela_distr.kurtosis	4	x		x	x	x	
ela_distr.number_of_peaks	4	x			x	x	x
ela_meta.lin_simple.adj_r2	2	x					x
ela_meta.lin_simple.intercept	1	x					
ela_meta.lin_simple.coef.min	2	x					x
ela_meta.lin_simple.coef.max	2	x		x			
ela_meta.lin_simple.coef.max_by_min	3				x	x	x
ela_meta.lin_w_interact.adj_r2	1	x					
ela_meta.quad_simple.adj_r2	2	x		x			
ela_meta.quad_simple.cond	4	x			x	x	x
ela_meta.quad_w_interact.adj_r2	3	x		x			x
disp.ratio_mean_02	1	x					
disp.ratio_mean_05	1	x					
disp.ratio_mean_10	1	x					
disp.ratio_mean_25	1	x					
disp.ratio_median_02	2	x					x
disp.ratio_median_05	1	x					
disp.ratio_median_10	1	x					
disp.ratio_median_25	1	x					
disp.diff_mean_02	1	x					
disp.diff_mean_05	1	x					
disp.diff_mean_10	1	x					
disp.diff_mean_25	1	x					
disp.diff_median_02	3	x				x	x
disp.diff_median_05	1	x					
disp.diff_median_10	1	x					
disp.diff_median_25	1	x					
nbc.nn_nb.sd_ratio	2	x					x
nbc.nn_nb.mean_ratio	1	x					
nbc.nn_nb.cor	2	x					x
nbc.dist_ratio.coeff_var	3	x				x	x
nbc.nb_fitness.cor	2	x		x			
ic.h.max	3	x				x	x
ic.eps.s	1	x					
ic.eps.max	1	x					
ic.eps.ratio	4	x		x		x	x
ic.m0	3	x				x	x
step_size	4	x			x	x	x
mahalanobis_dist	2				x		x
c_evol_path	3				x	x	x
sigma_evol_path	2					x	x
cma_simil_lh	2	x					x

Table 3. Feature portfolios.