Explorative data analysis of time series based algorithm features of CMA-ES variants
Nobel, J.P. de; Wang, H.; Bäck, T.H.W.; Chicano, F.

Citation
Nobel, J. P. de, Wang, H., & Bäck, T. H. W. (2021). Explorative data analysis of time series based algorithm features of CMA-ES variants. Gecco ’21: Proceedings Of The Genetic And Evolutionary Computation Conference, 510-518. doi:10.1145/3449639.3459399

Version: Publisher's Version
License: Leiden University Non-exclusive license
Downloaded from: https://hdl.handle.net/1887/3277010

Note: To cite this publication please use the final published version (if applicable).
We conducted a hierarchical clustering analysis on the test problem contrived to predict those measured values on each test problem. Also, it is known that their empirical performance is often hard to distinguish either CMA-ES variants or the problems. After measuring the number of predefined targets reached by those variants, we contrive to predict those measured values on each test problem using the feature selection procedure. From our analysis, we saw that the features can classify the CMA-ES variants, or the function groups decently, and show a potential for predicting the performance of those variants. We conducted a hierarchical clustering analysis on the test problems and noticed a drastic change in the clustering outcome when comparing the longer cutoff time to the shorter one, indicating a huge change in search behaviour of the algorithm. In general, we found that with longer time series, the predictive power of the time series features increase.

1 INTRODUCTION

In the field of continuous black-box optimization, a large variety of optimization algorithms are proposed in the past [7, 8] and some prominent ones, e.g., the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [15] have been successfully applied to solve real-world problems. More recently, many interesting scientific efforts have been devoted to these algorithms to boost up their performance even further. Particularly, for the CMA-ES, many variants have been devised based on the standard algorithm [15], e.g., the so-called BIPOP-CMA-ES [11] (please see [28] for an overview). Also, it is known that their empirical performance is often hard to distinguish with a statistical meaning when aggregated over a large number of different test problems [28]. This given, provides the need to tackle the well-known algorithm selection problem [24], which targets to find a proper optimization algorithm among a portfolio on some problem class [20]. As an effective approach into this direction, Explorative Landscape Analysis (ELA) [19] claims that it is vital for the algorithm selection task to obtain knowledge on optimization problems. It contrives to acquire such knowledge on the properties of test problems by designing numerical features. Rather than focusing on the problem side, in this paper, we argue that it is equally important to construct a reliable and useful quantification of algorithm behaviour. Answering the question; how different optimization algorithms would behave on different optimization problems since conceivably, an appropriate algorithm could be chosen more effective if we have sufficient knowledge from both the algorithm and problem side.

Here, the first step in this direction is taken by monitoring the strategy parameters of the CMA-ES variants during the optimization runs and extracting time series features from the dynamics of those parameters. Specifically, we adopted a wide set of pre-defined time series features as in the tsfresh library, e.g., the entropy or the absolute sum of changes of the series, and apply them on the strategy parameters of the CMA-ES, e.g., the step-size and the eigenvalues of the covariance matrix. For instance, we could compute the so-called "absolute sum of changes" feature for the step-size, which measures how much the step-size oscillates over time. In this manner, time series analysis could provide insight into how the different CMA-ES variants behave over time by quantifying the differences in their features.

Facilitated by this approach, we analyse the differences between several different CMA-ES variants on single or multiple test problems. We also delve into the dynamics of a single variant’s strategy parameters on different benchmark problems. This also provide us knowledge about the objective function a variant is optimizing, allowing us to classify objective functions in terms of algorithm dynamics. Furthermore, this could also open the possibility of quantifying problem properties, serving as as an alternative to the standard ELA methodology. We further extend our investigation by exploring the feasibility of predicting the performance of those variants using the its time series features.

As for the experimental study, we took the well-know BBOB test problems from the COCO platform [13] and conducted clustering analysis and predictive modeling for the time series features.

This paper is organized as follows; section 2 introduces the necessary preliminaries for the proposed time series feature based approach, described in section 3, including a hierarchical clustering experiment. Section 4 includes a supervised learning experiment with the computed time series features. This is followed by a regression approach in section 5, investigating the relation between...
the strategy parameters, which is why such variants are left out. The list includes:

(1) **Standard**: The standard CMA-ES, as defined in [15].
(2) **Active**: CMA-ES with active covariance matrix update [16] penalizes bad candidate solutions with negative weights in the covariance matrix update.
(3) **Mirrored sampling** [3]: For each new sampled point, a mirrored pair is produced.
(4) **Mirrored Pairwise sampling** [2, 29]: Mirrored sampling, however, only the best of the two mirrored points is selected.
(5) **Orthogonal** sampling: The sampled candidate solutions are orthonormalized using a Gram-Schmidt procedure [29].
(6) **Elitist**: \((\mu + \lambda)\) - selection.
(7) **Equal Weights**: Recombination weights of \(\frac{1}{2}\)
(8) **MSR**: Median success rule step size adaptation [1].
(9) **TPA**: Two-point step size adaptation [10].
(10) **Halton sampling**: Quasi-random Halton sampling [25].
(11) **Sobol sampling**: Quasi-random Sobol sampling [25].
(12) **Threshold Convergence**: Threshold Convergence of mutations [23]

CMA-ES’ dynamic parameters. Many internal parameters of the CMA-ES algorithms are adapted throughout the optimization process, e.g., the center of mass and the covariance matrix. Among all such parameters, we consider, in this work, the following ones for our analysis:

- \(\sigma \in \mathbb{R}_{\geq 0}\): The step size.
- \(\hat{f}_{\text{best}} \in \mathbb{R}\): The best-so-far objective value.
- \(\bar{\sigma} = \left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_d}\right)\): The square root of the diagonal of \(D\).
- \(\hat{p}_e \in \mathbb{R}^d\): The evolution path.
- \(\hat{p}_o \in \mathbb{R}^d\): The conjugate evolution path.

The covariance matrix \(C\) and the basis of its eigenvectors \(B\) are not considered in this experiment, as they would immensely increase the number of observations stored, while the information gained in storing these variables is small in comparison. Also, the center of mass \(\hat{m}\) is also not taken into account, as we consider it less of a variable telling something about the state of the algorithm than about a location in the search space.

**Time series features.** Here, we consider a multivariate real-valued time series with a fixed sampling frequency [4], which represents, in this context, the dynamic strategy parameters of the CMA-ES algorithm. In each iteration of CMA-ES, we compute one data point of this time series as follows: \(\forall t \in [L]\),

\[
\mathbf{x}_t = (\sigma, \hat{f}_{\text{best}}, ||\bar{\sigma}||, ||\hat{p}_e||, ||\hat{p}_o||, \text{ave}(\bar{\sigma}), \text{ave}(\hat{p}_e), \text{ave}(\hat{p}_o))^T ,
\]

where \(||\bar{\sigma}|| = \sqrt{\bar{\sigma}^T \bar{\sigma}}\) and \(L\) is the total number of iterations (running length) executed in our experiments and \(\text{ave}(\cdot)\) stands for the average of the components of its argument. We compute the component-wise average and norm of each vector variable in order to store their information invariant to problem dimension. Hereafter, we shall denote this multivariate time series as \(X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L] \in \mathbb{R}^{d \times L}\).

Given the time series data, we often probe into the characteristic thereof by means of extracting numerical features in the time domain, which give rise to meaningful interpretations and also embed the high-dimensional time series data into a lower-dimensional
\[ X \mapsto (\phi_1(X_{i1}), \ldots, \phi_1(X_{i5}), \ldots, \phi_m(X_{i1}), \ldots, \phi_m(X_{i5}))^T, \]

where \( X_{i} \) represents the \( i \)th row of \( X \). Hereafter, we shall denote a feature vector as \( \Phi(X) \), or simply \( \Phi \) if the input time series is clear from the context. The feature functions applied in our experiment can be found in table 2. When applied in our case, such a feature extraction procedure, results in 6,352 features (see section 3.1 for detail), which ought to undergo a proper feature selection process (see section 3.2) to reduce its dimensionality, hence facilitating the modelling tasks later.

3 THE PROPOSED APPROACH

Given a benchmarking problem set \( \mathcal{F} = \{f_1, f_2, \ldots\} \), a set of CMA-ES variants \( \mathcal{A} = \{A_1, A_2, \ldots\} \), and a set of time series feature functions, we aim to distinguish those variants for each \( f \in \mathcal{F} \) by computing the aforementioned time series features of every variant on \( f \) and building a classifier based on the features. Specifically, we repeat the feature computation \( N \) times independently for each \( A \) and \( f \), since the time series features from a single run exhibit large variability. We shall denote by \( \Phi(A, f, d, i) \) the time series features computed from the \( i \)th independent run of a variant \( A \) on the \( d \)-dimensional function \( f \). Taking this notation, we could express the data set for classifying (see section 4) CMA-ES variants as \( \{(\Phi(A, f, d, i), A)\}_{i=1}^{N}, A \in \mathcal{A}, f \in \mathcal{F} \) and similarly the one for discriminating test problems, \( \{(\Phi(A, f, d, i), f)\}_{i=1}^{N}, A \in \mathcal{A}, f \in \mathcal{F} \). As previously mentioned, we consider 12 CMA-ES variants for \( A \), 24 BBOB problems for \( \mathcal{F} \) with \( N = 100 \) and \( d = 5 \). For all the classification tasks presented in the sequel, we use a variant of the random forest model, called Extremely Randomized Trees Classifier (ERTC) [9], which constructs each constituting decision tree without bootstrapping based on random splitting of internal nodes (implemented by the Python library scikit-learn [22]). We took the default hyper-parameter setting of this algorithm since it is not the focus of this work to perform a hyperparameter tuning task and the default hyperparameter settings are empirically robust. We choose this model because it treats all features more equally compared to the random forest, since an aggressive feature selection was applied before training this model (see Sec. 3.2).

3.1 Data generation

To generate the time series data, we conducted \( N = 100 \) independent runs for each pair of CMA-ES variant and test function in 5D, using the first instance \( (i \text{id} = 1) \) of each problem. In addition, for each run, we start all CMA-ES variants with the same initial mean, step-size, and covariance matrix (set to the identity matrix) to remove randomness incurred by the initialization, making the time series yielded from different variants/problems more comparable. Each independent run was terminated with a fixed number of 500 generations, from which two sets of features were extracted, one using only the first 100 generations (represented by \( L = 100 \)) of the data and the other using the complete length of \( 500 \) generations \( (L = 500) \). In detail, we utilized the tsfresh [5] python library (version 0.16) to extract the features of time series \( X \), where we took all 794 default features\(^4\), resulting in \( 794 \times 8 = 6352 \) raw feature values after applied over each time series.

Due to the fact that different feature values vary largely from one feature to the other and also from functions to functions, we re-scale the feature value to the unit interval for each time series on each function. Besides, we also recorded the number of predefined targets.

\(^4\)This huge set of features are obtained from the combination of 63 default feature functions and several different parameterizations for each of them. Please see https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html for the detail.
$N_t = \left\{ 10^2-(x-1)/5 : x \in [0, .51] \right\}$ being hit in each independent run, for the purpose of predicting the performance of each CMA-ES variants based on its time series features.

### 3.2 Feature selection

The vast number of raw features cast any learning task that takes them as input as a high-dimensional problem, and hence it is necessary to reduce its dimensionality by selecting more informative features. For this purpose, we picked the Boruta\cite{26} algorithm\footnote{It is available in Python: https://github.com/blue-yonder/tsfresh.}, which works in conjunction with the Random Forests Classifier (RFC) and tries to capture all the important features in the data set with respect to a target variable (here: the CMA-ES variant label). It creates so-called shadow features by randomly permuting the values for each feature, then trains a random forest\footnote{Essentially, any other feature importance calculation can also be applied here, e.g., from Gradient Boosting.} on the combination of the original features and the shadow features, and finally determines to retain a feature if its feature importance is significantly higher than its shadow counterpart. We took the default parameter settings of Boruta except that we set its maximal number of iterations to 150 and took the “auto”\footnote{Please see https://github.com/scikit-learn-contrib/boruta_py for more information on the Boruta package} option for the number of trees of the random forest model employed therein. Practically, we grouped the feature data by unique pairs of test problems and two running lengths $L \in \{100, 500\}$, which leads to 48 groups, and performed the feature selection procedure on each group separately with Boruta, where a random forest model is trained to classify the algorithm variants on each test problem based on their selected time-series features.

We only keep features that are selected by Boruta in all aforementioned 48 groups of data. If multiple parameterizations of the same feature and time series combination are selected, only the combination that shows the highest feature importance is retained. The feature importance is retrieved from the random forest model underpinning the Boruta algorithm. This selection procedure managed to reduce the dimensionality drastically from 5352 to 32, which is summarized in table 1 in terms of the number of selected ones for each raw time series, accompanied by the total and average feature importance (FI) values from the random forest model employed in Boruta. From the table, we can observe that 1) none of the features originating from $f_{test}$ is selected; 2) features from $||\vec{p}_r||$ are selected the most frequently; 3) features based on $\sigma$ show the highest average feature importance. Table 2 shows a specific overview of the selected feature functions and their parameter sets, as well as the time series they were applied over.

Furthermore, we visualize the selected features of all test problems in Fig. 1, which is obtained by embedding the features to a two-dimensional Euclidean space using the t-SNE\cite{26} dimensionality reduction technique. From the figure (left), we can see that many variants form distinct clusters in this embedded space, while some variants (e.g., Standard and Mirrored-Pairwise), however, are hardly distinguishable from each other. This is to be expected, since some variants are functionally more similar than other variants.

### 3.3 Clustering Analysis

Here, we propose to measure the similarity between CMA-ES variants by taking the mean of corresponding feature vectors over all test problems and runs, and then computing the mean Euclidean distance among the resultant. We render the distance matrix as a heatmap in Fig. 3 for both $L = 100$ (the lower triangle) and $L = 500$ (the upper triangle), which we observe that TPA and Threshold Convergence are the most dissimilar pair of variants, which are also quite distant from the other variants. For $L = 100$, the most similar ones are the Sobol and Halton variants. Even though they still show a very high similarity for $L = 500$, the most similar pair is the Standard CMA-ES and Mirrored Sampling with pairwise selection.

| Series | #selected features | Total FI | Average FI |
|--------|--------------------|----------|-------------|
| $||\vec{p}_r||$ | 9 | 1.1071 | 0.1230 |
| $\text{ave}(\vec{p}_r)$ | 1 | 0.0398 | 0.0398 |
| $||\vec{p}_c||$ | 8 | 0.6081 | 0.0760 |
| $\text{ave}(\vec{p}_c)$ | 0 | - | - |

Table 1: Grouped by the time series, the number of selected features and their corresponding feature importance are listed with the highest value depicted in boldface.

| Feature Function | Applied Over | Parameters |
|------------------|--------------|------------|
| absolute sum of changes | $||\vec{p}_r||$ | 0 |
| approximate entropy | $||\vec{p}_r||$ | $\{12, 1\}$ |
| autocorrelation | $||\vec{p}_r||$ | $\{1, 1\}$ |
| change_quantiles | $||\vec{p}_r||, \text{ave}(\vec{p}_r), ||\vec{p}_c||, \sigma$ | $\{\text{mean}, \{\text{F, 5, 6, 2}\}, \text{mean}, \{\text{F, 4, 6, 0}\}$ |
| cid_ce | $||\vec{p}_r||$ | $\{1, 1\}$ |
| energy_ratio_by_chunks | $||\vec{p}_r||, \text{ave}(\sigma)$ | $\{10, 0\}$ |
| fft_aggregated | $||\vec{p}_r||$ | $\{\text{centroid}\}$ |
| fft_coefficient | $||\vec{p}_r||$ | $\{0, \text{abs}\}$ |
| index_mass_quantile | $||\vec{p}_r||, \text{ave}(\sigma)$ | $\{1, 1\}$ |
| mean | $||\vec{p}_r||$ | 0 |
| median | $||\vec{p}_r||, \text{ave}(\sigma), ||\vec{p}_c||, ||\vec{p}_r||$ | 0 |
| minimum | $||\vec{p}_r||, \text{ave}(\sigma)$ | 0 |
| number_of_crossing_m | $||\vec{p}_r||$ | $\{1, 1\}$ |
| number_peaks | $\sigma$ | $\{1, 1\}$ |
| partial_autocorrelation | $||\vec{p}_r||$ | $\{1, 1\}$ |
| quantile | $||\vec{p}_r||, \text{ave}(\sigma), ||\vec{p}_c||, ||\vec{p}_r||, \sigma$ | $\{1, 1\}$ |
| range_count | $||\vec{p}_r||, ||\vec{p}_c||$ | $\{11, -1, 1\}$ |
| sum_values | $||\vec{p}_r||$ | 0 |

Table 2: The selected time series features shown as the combination of the feature function, the time series on which the feature function is applied, and the parameterization therein. We express the parameterization as a set of tuples (empty for the parameterless case), where each tuple of parameter values is only applied on the time series appearing in the same order as with the tuple, e.g., for the “change_quantile” function, $\{\text{mean}, \{0, 2, 6, 0\}\}$ is only applied on $\sigma$. Please see https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html for the detail.
Figure 2: The hierarchical clustering results obtained on the pairwise distance of CMA-ES variants (left) and of test problems (right), in which we also contrast the $L = 100$ case with $L = 500$ to demonstrate the extent to which the clustering result would be affected by a prolonged running length. Also, we take the distance between the cluster to determine the color scale for the line connecting them. The dendrograms were produced for the mean vector of each variant/test function, using Euclidean distance and the Ward [21] hierarchical clustering.

In addition, we ran a more fine-grained hierarchical clustering on the distance matrix (shown in Fig. 2, left). The similarities between the variants invoked the following consideration: variants which are seemingly different in terms of algorithmic structure, can exhibit similar behaviour. This suggests that it can be helpful to probe into the dynamics of newly designed variants using time-series features, if such a variant does not perform as expected.

The distance among all test problems is only computed from features of the standard CMA-ES in line with the classification setup (see section 4). In Fig. 4 a pairwise distance matrix is showing this result. We also drawn the result of a hierarchical clustering in Fig. 2 (on the right). From those two figures, we can see that the most closely related functions seem to be the triplet of functions $f_3, f_4,$ and $f_{20}$, which is also very close to $f_{15}, f_{16}, f_{17},$ and $f_{18}$ and hence forms a bigger cluster together. Besides, problem $f_1$ and $f_{21}$ are also decently similar, opening an interesting question on interpreting it since $f_1$ is generally considered simple while $f_{21}$ is conceivably difficult. We hypothesize that this is because the local topology of $f_{21}$ resembles the global topology of $f_1$.

Another intriguing observation is that problem $f_{13}$ is noticeably different from the other problems for $L = 500$ while such a difference does not exists for the $L = 100$ case. This indicates that the dynamics of the standard CMA-ES when optimizing $f_{13}$ are similar to the other functions in the first 100 generations of the optimization procedure and it that differentiates substantially from the others in the next 400 generations. This entails that there might be huge changes to the landscape of $f_{13}$ between generations 100 and 500, attributing to this huge change in the CMA-ES’s behavior. Interestingly, this connotation matches the design rationale behind $f_{13}$ [14]: “Approaching the ridge is initially effective, but becomes ineffective close to the ridge... The necessary change in ‘search behavior’ close to the ridge...”, hinting that a solver needs to change its search behaviour as it approaches the ridge defined in the function landscape.

Figure 3: Pairwise distance matrix for all CMA-ES variants. The numeric index used here refers to the number specified in the list in section 2. Euclidean distance is computed and scaled though min-max normalization. The lower triangle of the figure shows the pairwise distances for $L = 100$, the upper triangle shows $L = 500$.
we observe an average accuracy of \( \frac{1}{7} \) on problem \( f_1 \) and the classification performance. In detail, the lowest accuracy is \( \frac{1}{7} \) for both \( L = 100 \) and \( L = 500 \). For the former scenario, we perform a K-fold CV (with \( K = 24 \)) and summarize the resulting classification accuracy in table 3, in which we observe an average accuracy of \( 0.803 \pm 0.087 \) for \( L = 100 \) and \( 0.937 \pm 0.069 \) for \( L = 500 \), indicating that a longer time series improves the classification performance. In detail, the lowest accuracy is observed on problem \( f_3 \) for both \( L = 100 \) and \( L = 500 \) while the highest one is achieved on \( f_1 \) for \( L = 100 \) and \( f_6 \) for \( L = 500 \).

Figure 4: Pairwise distance matrices for all functions using data from the Standard CMA-ES. Euclidean distance is computed and scaled though min-max normalization. The lower triangle of the figure shows the pairwise distances for \( L = 100 \), the upper triangle shows \( L = 500 \).

### 4 CLASSIFYING CMA-ES VARIANTS AND PROBLEMS

Taking the 32 selected features, we constructed Extremely Randomized Trees Classifiers (ERTCs) to discriminate CMA-ES variances and the test problems for both cases when \( L = 100 \) and \( L = 500 \). This allows for investigating the impact of the length of raw time series, in terms of the classification power. Also, the resulting models are assessed using cross validation (CV) strategy (please see below for the detail). In addition to model accuracy we measure F1-score. Specifically, we calculate, for each fold, the precision (\( p \)) and recall (\( r \)) with respect to each label and take the harmonic mean, i.e., \( \frac{2}{1/p + 1/r} \).

Classifying CMA-ES variants. Here, we consider a multi-class classification task to distinguish CMA-ES variants on

- each test problem separately, in which we have \( |A| \times N = 1200 \) data points in each case, and
- on the whole data set from all 24 test problems.

For the former scenario, we perform a K-fold CV (with \( K = 24 \)) and summarize the resulting classification accuracy in table 3, in which we observe an average accuracy of \( 0.803 \pm 0.087 \) for \( L = 100 \) and \( 0.937 \pm 0.069 \) for \( L = 500 \), indicating that a longer time series improves the classification performance. In detail, the lowest accuracy is observed on problem \( f_3 \) for both \( L = 100 \) and \( L = 500 \) while the highest one is achieved on \( f_1 \) for \( L = 100 \) and \( f_6 \) for \( L = 500 \).

Table 3: For classifying CMA-ES variants on each test problem separately, we list the accuracy and F1 scores resulted from a 24-fold CV procedure with the highest measured values marked in boldface and the lowest ones underlined.

| FID  | Acc. \( L = 100 \) | Acc. \( L = 500 \) | F1 \( L = 100 \) | F1 \( L = 500 \) |
|------|-------------------|-------------------|----------------|-----------------|
| \( f_1 \) | 0.939 ± 0.032 | 0.991 ± 0.014 | 0.939 ± 0.034 | 0.991 ± 0.014 |
| \( f_2 \) | 0.742 ± 0.065 | 0.980 ± 0.022 | 0.741 ± 0.071 | 0.980 ± 0.022 |
| \( f_3 \) | 0.847 ± 0.057 | 0.983 ± 0.018 | 0.843 ± 0.059 | 0.984 ± 0.018 |
| \( f_4 \) | 0.821 ± 0.050 | 0.983 ± 0.018 | 0.819 ± 0.050 | 0.983 ± 0.018 |
| \( f_5 \) | 0.793 ± 0.064 | 0.822 ± 0.070 | 0.789 ± 0.072 | 0.825 ± 0.075 |
| \( f_6 \) | 0.844 ± 0.053 | 0.995 ± 0.009 | 0.842 ± 0.054 | 0.995 ± 0.009 |
| \( f_7 \) | 0.829 ± 0.050 | 0.878 ± 0.054 | 0.829 ± 0.055 | 0.877 ± 0.058 |
| \( f_8 \) | 0.771 ± 0.062 | 0.971 ± 0.024 | 0.764 ± 0.062 | 0.971 ± 0.024 |
| \( f_9 \) | 0.788 ± 0.066 | 0.982 ± 0.022 | 0.782 ± 0.067 | 0.982 ± 0.024 |
| \( f_{10} \) | 0.771 ± 0.042 | 0.986 ± 0.015 | 0.765 ± 0.046 | 0.986 ± 0.015 |
| \( f_{11} \) | 0.796 ± 0.076 | 0.992 ± 0.013 | 0.795 ± 0.080 | 0.992 ± 0.013 |
| \( f_{12} \) | 0.877 ± 0.039 | 0.945 ± 0.020 | 0.877 ± 0.040 | 0.946 ± 0.019 |
| \( f_{13} \) | 0.828 ± 0.051 | 0.881 ± 0.043 | 0.826 ± 0.058 | 0.882 ± 0.041 |
| \( f_{14} \) | 0.870 ± 0.044 | 0.935 ± 0.037 | 0.868 ± 0.043 | 0.936 ± 0.036 |
| \( f_{15} \) | 0.809 ± 0.051 | 0.975 ± 0.027 | 0.812 ± 0.053 | 0.976 ± 0.027 |
| \( f_{16} \) | 0.800 ± 0.051 | 0.944 ± 0.026 | 0.804 ± 0.052 | 0.944 ± 0.027 |
| \( f_{17} \) | 0.788 ± 0.057 | 0.878 ± 0.047 | 0.788 ± 0.056 | 0.878 ± 0.048 |
| \( f_{18} \) | 0.748 ± 0.057 | 0.850 ± 0.056 | 0.748 ± 0.061 | 0.846 ± 0.055 |
| \( f_{19} \) | 0.680 ± 0.052 | 0.845 ± 0.052 | 0.681 ± 0.057 | 0.842 ± 0.051 |
| \( f_{20} \) | 0.834 ± 0.049 | 0.978 ± 0.016 | 0.834 ± 0.051 | 0.979 ± 0.016 |
| \( f_{21} \) | 0.958 ± 0.033 | 0.992 ± 0.021 | \textbf{0.958 ± 0.033} | 0.992 ± 0.021 |
| \( f_{22} \) | 0.767 ± 0.049 | 0.939 ± 0.034 | 0.767 ± 0.052 | 0.941 ± 0.033 |
| \( f_{23} \) | 0.677 ± 0.072 | 0.810 ± 0.063 | 0.670 ± 0.077 | 0.807 ± 0.066 |
| \( f_{24} \) | 0.705 ± 0.066 | 0.952 ± 0.035 | 0.704 ± 0.070 | 0.953 ± 0.037 |

Avg. \( 0.803 ± 0.067 \) \( 0.937 ± 0.058 \) \( 0.802 ± 0.068 \) \( 0.937 ± 0.059 \)

For the latter scenario, we merged data points from all 24 test problems and conducted two different CV strategies: a standard randomized K-fold CV with \( K = 24 \) and a Leave-One-Group-Out (LOGO) CV, where we hold data points from a single problem for validation and train the model on the remaining 23 problems. Note that the LOGO procedure iterates over each test problem. We show the resulting accuracy and F1 score in table 4, in which we see that compared to both performance values in the K-fold case, the ones from LOGO CV drop slightly. This matches our expectation that the model would deteriorate if we exclude entirely the information that whether the time series features are capable of discrimination, e.g., on \( f_{12} \), \( f_{13} \), and \( f_{14} \), \( L = 100 \) it is noticeably higher.

Classifying test problems. Furthermore, we delve into the question that whether the time series features are capable of discriminating test problems, which, if so, could serve as an alternative to the well-known Explorative Landscape Analysis (ELA) features [17] for identifying landscape classes. For this purpose, we took only the
we perform classification for both cases when 

Table 4: Comparison of two classification approaches for 

variants. The fact that \( f \) very similar classification results. Hence, it is more natural to take the baseline variant. 

other functions. This then also means, that when using data of 

pattern opposite of that of figure 5. Now, 

time giving a correct prediction, because it is now working on data 

model trained on the remaining 23 problems. The 

patterns in the data, it shows lower performance in classifying 

for this experiment, which shows the F1-score per classification 

observed. This is in line with our earlier observation from the LOGO 

experiment, rather poor classification performance was 

\( f \) functions to predict, while when using data from 

Explorative Data Analysis of Time Series based Algorithm Features of CMA-ES Variants 

GECCO ’21, July 10–14, 2021, Lille, France 

time series feature data from the Standard CMA-ES to distinguish 

test problems using the ERTC method\(^2\). As with the previous task, 

we perform classification for both cases when \( L = 100 \) and \( L = 500 \), and use 24-fold cross validation. 

This experiment, however, only achieves an accuracy of \( .497 \pm .053 \) for the \( L = 100 \) while it manages to reach \( .716 \pm .048 \) for \( L = 500 \). In contrast to the study on classifying algorithms, we encounter 

a substantial performance drop. In figure 6, a box plot is shown 

for this experiment, which shows the F1-score per classification 

label obtained for each cross validation fold. Again, we observe 

a large variance between the labels. Interestingly, we observe a 

pattern opposite of that of figure 5. Now, \( f_5 \) is one of the easiest 

functions to predict, while when using data from \( f_5 \) for testing in the 

LOGO experiment, rather poor classification performance was 

observed. This is in line with our earlier observation from the LOGO 

experiment; when a classifier is not able to learn function specific 

patterns in the data, it shows lower performance in classifying 

variants. The fact that \( f_5 \) is classified correctly more often than 

most other functions in this experiment, means that there is a 

pattern in the data of \( f_5 \) which distinguishes it from the data of 

other functions. This then also means, that when using data of 

\( f_5 \) for testing in the LOGO experiment, the classifier has a harder 

time giving a correct prediction, because it is now working on data 

which is very dissimilar from the data is was trained on. The reverse 

can also be observed in \( f_5 \). Because there are other functions in 

the data which express similar patterns, such as \( f_5 \), the classifier 

is not able to clearly distinguish between them. However, when 

predicting variants (figure 5), the classifier is able to learn these 

function specific patterns during training from \( f_4 \), which in turn 

allows it to more confidently predict variants when using data taken 

from \( f_5 \) for testing. 

In addition to classify each test problem, we perform a classification 
task on problem groups, where the problems are partitioned according to the default groups in COCO/BBOB, or to the clustering results in [20] (referred as Mersmann’s grouping hereafter), which is obtained on the expected running time (ERT) values of 51 different optimization algorithms on SD. The latter grouping might be particularly suitable for this study, since it is obtained based on the consensus rankings of different optimization algorithms, which reflects, to some extent, the algorithmic behaviour and hence could be well related to the time series features extracted from dynamics of the algorithm. This conjecture is empirically supported by table 5, where the classification outcome of Mersmann’s grouping is much better than that of the default BBOB grouping for \( L = 100 \) and \( L = 500 \).

5 PREDICTING CMA-ES’S PERFORMANCE

From figure 1 (right), we see clearly that feature vectors that are 
close to each other in terms of its two-dimensional embedding 
appear to be associated with similar numbers of targets reaches, 

which motivates us to predict the performance of each run using 

the corresponding features. For this goal, we adopted an Extremely 

Randomized Trees Regressor (ERTR, from scikit-learn [22]) with its 
default parameter setting. The independent variables in this regression 

experiment consist of the 32 selected time series features, which 

are taken as input to predict the number of targets \( N_t \). Also, we 
took the coefficient of determination \( R^2(y, \hat{y}) = 1 - \frac{\Sigma(y-\hat{y})^2}{\Sigma(y-\text{mean}(y))^2} \) 
as the performance metric and applied a 24-fold CV procedure as 
before, for both \( L = 100 \) and \( L = 500 \) cases.

When training the ETR model on all variants (on all problems), we observed an \( R^2 \) of \( .557 \pm .018 \) for \( L = 100 \), and \( .778 \pm .014 \) for 

\( L = 500 \). Alternatively, when training the model for each variants 

separately (see Fig. 7), we see the a huge improvement of the \( R^2 \) 
on the \( L = 500 \) case for most variants, which is also considerably 
stable in contrast to the \( L = 100 \) case. It is worth noting that we do 
not consider a scenario in which we took each problem separately, 
since the distribution of \( N_t \) value varies drastically from function 
to function. This is because the CMA-ES shows a different performance on each benchmark function, signifying that data from

\[^2\]This choice is made based on our observation that all CMA-ES variants give rise to

very similar classification results. Hence, it is more natural to take the baseline variant.
Figure 6: On classifying the test problems, we illustrate the resulting F1-score per each classification label, where the box plot indicates the variability of this score across a 24-fold cross validation. We also contrast the $L = 100$ case with the $L = 500$ one.

Figure 7: Regression, with a model trained on each CMA-ES variant separately, showing a box plot of the $R^2$ attained using K-fold cross validation, with K=24. The numeric index used here on the x-axis refers to the number specified in the list in section 2. Data for $L = 100$ is shown in blue, $L = 500$ is shown in orange.

6 CONCLUSIONS

In this work, we have used time series characterization methods to extract features from the dynamic strategy parameters of the CMA-ES variants during the optimization of 24 test problems from the Black-Box Optimization Benchmark (BBOB). After extracting the time series features on 5D with two different running length of each variant ($L = 100$ and $L = 500$ for 100 and 500 iterations, respectively), we conducted a proper feature selection to reduce the dimensionality of the resulting features. We built extremely randomized trees to classify the variants on a single and multiple problems, as well as to distinguish those test problems. Furthermore, we performed in-depth explorative data analysis on the selected features, where we show the hierarchical cluster discovered in CMA-ES variants and in test problems and contrast such clustering results obtained in both $L = 100$ and $L = 500$ cases. Moreover, we explored the potential of directly predicting the performance of CMA-ES variants using their corresponding features.

We observed that the time series feature are capable of discriminating CMA-ES variants with nearly perfect accuracy, if we train the model on a single problem only. We also managed to achieve a decent accuracy when classifying the function groups. When it comes to predict the performance of those variants, we saw an indication that the feature might be reasonably related to the performance, which is subject to future researches.

From the clustering analysis of test problems, we observed an intriguing pattern on problem $f_{13}$, which resembles many other problems for the $L = 100$ case and becomes very distant from all other functions for $L = 500$. This phenomenon suggests that there might be a drastic change in the landscape which causes the change of the behavior of the CMA-ES, which is reflected by the feature. This is also supported by the fact that the landscape of $f_{13}$ is very heterogeneous.

For the future research, we are planning to 1) apply this approach on more problems and algorithms; 2) to investigate the usefulness of the time series feature for the dynamic algorithm selection task; 3) to extend it by computing the features in a moving window; and 4) to dive more deeply into the data to investigate the relative contribution and values of each feature with the help of, for instance the Shapley value.

REFERENCES

[1] Ouassim Ait Elhara, Anne Auger, and Nikolaus Hansen. 2013. A Median Success Rule for Non-Elitist Evolution Strategies: Study of Feasibility. GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference (07 2013). https://doi.org/10.1145/2463772.2463429

[2] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. 2011. Mirrored sampling in evolution strategies with weighted recombination. In Proceedings of the 13th annual conference on Genetic and evolutionary computation. ACM, 961–968.
