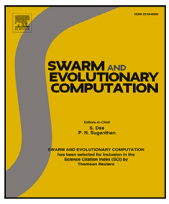
Swarm and Evolutionary Computation 98 (2025) 102116

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo



# Exploring module interactions in modular CMA-ES across problem classes

Ana Nikolikj [a,b,*], Tome Eftimov [a]

[a] Computer Systems Department, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, 1000, Slovenia
[b] Jožef Stefan International Postgraduate School, Jamova cesta 39, Ljubljana, 1000, Slovenia

ARTICLE INFO

*Keywords:*
Module importance
Empirical study
Black-box optimization
Benchmarking

ABSTRACT

This study presents an in-depth analysis of module importance within the modular CMA-ES (modCMA-ES) algorithm using exploratory data analysis and large-scale benchmarking across the BBOB suite. Rather than introducing new algorithms, our contribution lies in uncovering how individual modules and their interactions influence optimization performance across diverse black-box problem classes. We evaluate 324 modCMA-ES variants across 24 problem classes using functional ANOVA (f-ANOVA) to quantify the variance in performance attributable to individual, pairwise, and triplet module interactions. Results reveal substantial variation in module importance across problem classes and highlight strong alignment between module interaction patterns and high-level landscape features, particularly multi-modality. Further, we demonstrate that configuring only the most important modules — identified via f-ANOVA — achieves performance comparable to or better than the single-best solver, especially in high-dimensional settings. This analysis, conducted at both low (5D) and high (30D) dimensions, offers actionable insights into module interactions within the mod-CMA-ES framework.

## 1. Introduction

Black-box optimization focuses on the design and analysis of algorithms for problems where the objective function is either intractable or cannot be exploited, relying solely on sampling new candidate solutions and evaluating their quality iteratively. It encompasses both single-objective [1] and multi-objective problems [2–5], each presenting unique challenges in terms of convergence, diversity preservation, and performance evaluation.

Many advancements in the state-of-the-art continuous single-objective black-box optimization stem from variants of already established algorithms, rather than entirely new algorithmic paradigms. For example, the data from the Black-Box Optimization Benchmarking (BBOB) workshops [6] reveal over 50 distinct efforts on improving the performance of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), originally introduced in [7]. As the number of new mechanisms grows, it becomes crucial to systematically assess their importance for the performance of the algorithm.

In 2022, the BBO community highlighted major concerns regarding the widespread reliance on metaphor-driven metaheuristics [8]. Key criticisms include the use of uninformative metaphors [9,10], where proposed algorithms often lack scientific depth and contribute only marginally; the absence of genuine novelty [8,9,11–15], as many "new" algorithms replicate previously established ideas; and biased benchmarking practices [1,16,17], where comparisons are frequently made against outdated state-of-the-art methods or unrepresentative problem sets [18]. The common assessment of new mechanisms involves benchmarking a new variant against the original algorithm (i.e., the default variant) and a limited set of other variants. However, the implementation details of the variants can severely influence the behavior of an algorithm, hindering the process of fair comparison and interpretation of the comparative results [19]. Further, such comparisons completely ignore interactions with existing mechanisms - essential for accurately attributing the importance of the new mechanism [20]. Moreover, it disregards the underlying problem landscape properties, which are crucial for understanding the importance across different problem types [21,22]. Last, many variants are tailored to enhance performance on specific problems, often neglecting a more comprehensive evaluation across diverse problem types in the studies [23]. As a result, such a comparison contributes little to the general knowledge about the algorithm. These issues have led to an overabundance of redundant or superficial metaheuristics, generating confusion and hindering meaningful progress in understanding algorithmic behavior.

To address some of the above-mentioned issues, *modular algorithms* have been proposed, where the main idea is to break down an algorithm into smaller components called *modules* that can be combined and configured with different options, to create new algorithm variants. This allows us to investigate the contribution of newly invented mechanisms, rather than introducing a new one without a clear performance

* Corresponding author at: Computer Systems Department, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, 1000, Slovenia.
*E-mail address:* ana.nikolikj@ijs.si (A. Nikolikj).

https://doi.org/10.1016/j.swevo.2025.102116
Received 26 April 2025; Received in revised form 1 July 2025; Accepted 31 July 2025
Available online 14 August 2025

gain. There is already an adopted modular optimization framework - modCMA-ES - for the CMA-ES algorithm in study [24]. The mentioned study emphasizes the advantages of modular algorithms, both in terms of fair comparisons from an implementation perspective, and also structured analysis of module impact on the performance. However, the full potential of modular algorithms for quantifying module importance and module interactions remains unexplored.

A recent study [25] tackles this challenge by utilizing modular algorithms and explainable AI (XAI) techniques to gain insights into a broad set of modules. However, it focuses solely on the importance of individual modules, overlooking their interactions. In our previous study [26], we previously quantified interactions between modCMA-ES and modDE modules using f-ANOVA [27,28] across a benchmark suite of 24 problem classes, aiming to capture interaction effects across problems with diverse landscape characteristics. While that study briefly touched on problem-specific analyses — such as identifying problems with similar interaction patterns — the analysis was limited in scope and primarily served to highlight opportunities for future research, which we now pursue in this study. It is also important to emphasize that our earlier work focused solely on measuring the importance of interactions, without examining the specific module values driving those interactions or their relationship to problem landscape features.

*Our contribution:* We present an in-depth analysis of module importance for the modCMA-ES algorithm, by applying exploratory data analysis techniques in a large-scale benchmarking setting. The main contribution of our work lies not in introducing new algorithms or frameworks, but in offering novel insights into how existing algorithmic modules interact and influence performance across diverse black-box optimization problems. To investigate this, the study considers a configuration space of seven modules and the performance data from 324 modCMA-ES variants collected for the 24 problem classes from the BBOB benchmark suite [29]. Next, f-ANOVA is applied to quantify the variance in performance caused by individual modules, as well as caused by pairwise and triplet combinations of modules. In doing so, we address a critical gap in the literature on CMA-ES, offering an empirical perspective on how algorithmic modules collectively influence performance. The analysis is performed for each problem class in the BBOB suite. This allows us to find modules that are consistently important across a wide range of problem classes, making them important in general, and modules that are important in very specific cases. The analysis reveals high variations in module importance for different problem classes.

A key contribution of the current study is the alignment of clusters — comprising problem classes with similar module interaction patterns — with clusters based on predefined high-level landscape characteristics (e.g., multi-modality). The results show that the clusters align the most with the multi-modality categories and least with the separability categories for low dimensions, suggesting that module importance in modCMA-ES is primarily influenced by whether a problem class is multi-modal or uni-modal. In high dimensions, module importance is influenced by scalability, multi-modality, and global structure.

Finally, to validate whether the module importance derived from f-ANOVA holds enough useful information, we perform algorithm configuration by configuring the most important modules using their results, while keeping the options for the remaining modules at their default values. This is performed for each problem class separately. The variants are then tested on new problem instances from corresponding BBOB problem classes, and compared to a few baselines for algorithm configuration. We show that configuring only the most important modules identified by f-ANOVA yields performance comparable to the single-best solver in low dimensions and surpasses it in high dimensions.

The analysis is conducted twice, for low-dimensional problems with five dimensions ($d = 5$), and high-dimensional problems with thirty dimensions ($d = 30$), to contrast how the hyperparameter importance potentially changes with the problem complexity.

The insights gained from our analysis regarding module importance and the identification of optimal module options can serve as a valuable resource for various downstream tasks, such as informed selection of algorithm variants in practice, guided algorithm configuration tools, and enhanced performance prediction models.

**Outline:** Section 2 reviews the background on modular optimization frameworks, f-ANOVA, and summarizes previous research on module importance for the CMA-ES algorithm, together with studies from related domains that utilize f-ANOVA. Section 3 briefly revisits the methodology employed for the analysis. Section 4 outlines the experimental setup. Section 5 presents the results. In Section 6, we discuss the strengths and limitations of the analysis with directions for future research. Finally, Section 7 concludes the paper.

## 2. Background

This section introduces the necessary background on modular optimization frameworks, f-ANOVA, and summarizes previous research on module importance.

### 2.1. Modular optimization frameworks

The implementation details of an algorithm variant can severely influence its behavior. Modular optimization frameworks have been proposed to ensure consistent implementation across common mechanisms of different algorithm variants. The main idea is to break down an algorithm into smaller independent mechanisms called *modules*, each of which can be configured with different options to modify the algorithm's behavior [30]. Moreover, the independence of each module allows for its removal or alteration without compromising the overall functionality of the algorithm. This modular structure enables the straightforward integration and combination of modules to create new algorithm variants, with consistent implementation details.

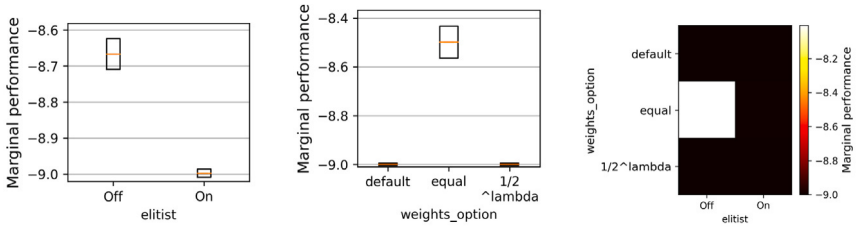
### 2.2. Functional ANOVA (f-ANOVA)

f-ANOVA relies on the concept of *marginal performance*, which refers to the average performance of a specific option for a module, computed across all possible combinations of the other modules. In simpler terms, it measures how well a particular choice for one module performs on average, regardless of the options chosen for the other modules. This helps isolate the contribution of that specific option to the overall performance.

Once the marginals (i.e., as many as options) for a module are computed, the variance in performance $V_i$ that module $i$ causes can be calculated with respect to the global mean performance across all variants. Intuitively, the higher the variance, the more important the module or the module combination. The importance of module $i$ is then defined as $I_i = \frac{V_i}{V_0}$ where $V_0$ is the total variance in the algorithm's performance. This normalization ensures that the importance values represent the fraction of variance attributed to a module, they range between 0 and 1, and the total sum of all importance values (including interactions) is approximately 1, given by $I = \sum I_i + \sum_{i \neq j} I_{i,j} + \sum_{i \neq j \neq k} I_{i,j,k} + \cdots \approx 1$, where $I_i$ is individual module importance, $I_{i,j}$ pairwise interaction importance, and $I_{i,j,k}$ triple interaction importance, etc. We refer to Hutter et al. [28] for a complete description.

The calculation of the marginals requires averaging performance over an exponentially large number of variants, making direct computation infeasible. The f-ANOVA framework addresses this by using a tree surrogate regression model that makes predictions for the performance of variants without the need to explicitly evaluate them. Surrogate models operate at the meta-level, taking as input module options for a certain variant and predicting its performance. The calculation of the marginal performance then scales linearly with the number of leaves in the tree regression model [28].

Just for illustration, Fig. 1 displays the marginals for two individual modules of the modCMA-ES algorithm and their pairwise interaction effect on the first problem class ($f\_id = 1$) in the BBOB suite. Further details of the outcomes will be presented in Section 5.




(a) *elitist* (b) *weights_options* (c) pairwise interaction

**Fig. 1.** Marginal performance of *modCMA-ES* module options on the first BBOB problem class in dimension $d = 5$. Performance is measured as the distance to the optimum and displayed on a $\log_{10}$ scale, where lower values indicate better results. Sub-figures (a) and (b) show the individual effects of the *elitist* and *weights_options* modules, respectively, while (c) illustrates their pairwise interaction. Setting *elitist* to *On* and using *default* or $(1/2)^\lambda$ for *weights_options* yields better performance individually. In combination, the choice of *weights_options* has a stronger influence, while *elitist* has minimal effect, with both its options having similar marginal performance.

**Table 1**
Classification of the noiseless BBOB problem classes based on their high-level landscape features: multi-modality, global structure (gl. struc), separability, variable scaling, homogeneity (homog.), basin sizes, and global-to-local contrast (gl.-loc). Horizontal lines separate the initial five problem classes.

| f_id | Predefined groups | Multi-modality | gl. struc. | Separability | Scaling | homog. | Basins | gl.-loc. | Funnel |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Separable | None | None | High | None | High | None | None | Yes |
| 2 | Separable | None | None | High | High | High | None | None | Yes |
| 3 | Separable | High | Strong | High | Low | None | Low | Low | Yes |
| 4 | Separable | High | Strong | High | Low | High | med. | Low | Yes |
| 5 | Separable | None | None | High | None | High | None | None | Yes |
| 6 | Low or moderate conditioned | None | None | High | Low | med. | None | None | Yes |
| 7 | Low or moderate conditioned | None | None | High | Low | High | None | None | Yes |
| 8 | Low or moderate conditioned | Low | None | None | None | med. | Low | Low | Yes |
| 9 | Low or moderate conditioned | Low | None | None | None | med. | Low | Low | Yes |
| 10 | High conditioned and uni-modal | None | None | None | High | High | None | None | Yes |
| 11 | High conditioned and uni-modal | None | None | None | High | High | None | None | Yes |
| 12 | High conditioned and uni-modal | None | None | None | High | High | None | None | Yes |
| 13 | High conditioned and uni-modal | None | None | None | Low | med. | None | None | Yes |
| 14 | High conditioned and uni-modal | None | None | None | Low | med. | None | None | Yes |
| 15 | Multi-modal problems with adequate gl. struc. | High | Strong | None | Low | High | Low | Low | Yes |
| 16 | Multi-modal problems with adequate gl. struc. | High | med. | None | med. | High | med. | Low | None |
| 17 | Multi-modal problems with adequate gl. struc. | High | med. | None | Low | med. | med. | High | Yes |
| 18 | Multi-modal problems with adequate gl. struc. | High | med. | None | High | med. | med. | High | Yes |
| 19 | Multi-modal problems with adequate gl. struc. | High | Strong | None | None | High | Low | Low | Yes |
| 20 | Multi-modal problems with weak gl. struc. | med | Deceptive | None | None | High | Low | Low | Yes |
| 21 | Multi-modal problems with weak gl. struc. | med. | None | None | med. | High | med. | Low | None |
| 22 | Multi-modal problems with weak gl. struc. | Low | None | None | med. | High | med. | med. | None |
| 23 | Multi-modal problems with weak gl. struc. | High | None | None | None | High | Low | Low | None |
| 24 | Multi-modal problems with weak gl. struc. | High | Weak | None | Low | High | Low | Low | Yes |

*For the third BBOB problem class, separability was originally marked as none in the original table in [32]. However, we have updated it to high based on the mismatch with the initial five features and the definition of the function available on the COCO platform.

### 2.3. High-level landscape features

The problem classes in the BBOB suite are systematically categorized using expert knowledge into five groups, each representing high-level features of the fitness landscape: (a) separable problems ($f1-f5$), (b) low or moderately conditioned problems ($f6-f9$), (c) highly conditioned uni-modal problems ($f10-f14$), (d) multi-modal problems with clear global structure ($f15-f19$), and (e) multi-modal problems with weak global structure ($f20-f24$) [31]. The categorization was later refined in [32], resulting in seven high-level features—multi-modality, global structure, separability, scaling, homogeneity, basin sizes, and global-to-local contrast—to provide a more detailed characterization of the problem classes in the benchmark suite. Lastly, the funnel structure [33] was introduced as a high-level feature to categorize the benchmark problems. The high-level features are summarized in Table 1.

### 2.4. Related work

Running various variants of an algorithm can lead to varying performances on a given optimization problem. It is not always clear which modules or their interactions drive this variation, or which are the respective good or bad options for the modules. Several techniques for hyperparameter/module importance exist, such as forward selection [34], performance-influence models [35], f-ANOVA [28,36], ablation analysis [37], and frequency-based impact scores [24].

The initial work on adopting the modCMA-ES algorithm and analyzing module importance is presented in [38]. The analysis is facilitated through performance data of 4608 modCMA-ES variants resulting from a complete enumeration of options for 11 modules. The module's importance is quantified in two ways: through importance scores extracted from a Decision Tree and as the difference in mean performance across different options of a module - to assess individual module significance. Additionally, module interaction importance is quantified as the difference in the mean performance when both modules are enabled or disabled at the same time, while other option combinations (e.g., one module on and the other is off) are not considered to lead to limited insights about the interaction. Another study [24] explores a wide space of algorithm variants through means of algorithm tuning and identifies a set of "elite configurations" that perform best on a per-problem basis. Then, the module's individual importance is represented

as the module's frequency in the elite set. To also quantify module interaction importance, this process is repeated by iteratively adding modules to the space explored by the algorithm configuration tool. The module interaction importance is then reflected in the change in frequencies within the elite configurations. However, the studies do not provide a more systematic way of contrasting module importance between different problem classes, grouping them, nor do they investigate the correlation between module importance and problem properties.

There are also several studies studying the link between problem landscape features and the performance of different modules [39–42]. In [39], a classifier chain approach is used, where each module option is predicted by a base classifier that takes as input the outputs of all previous classifiers and the landscape features. Since each classifier uses the predicted options of previous modules, it implicitly learns how choices of some modules affect the selection or behavior of others. While this approach models interactions for predictive purposes, it does not explicitly quantify which interactions are most important, nor how module combinations impact performance. A recent study [42] leverages problem landscape features to train separate classifiers for predicting the optimal choice for each module of modCMA-ES. However, this does not account for interactions between the modules. The approach in [25] applies XAI techniques to analyze the importance of algorithm modules and their options. The study is based on Shapley values from game theory to evaluate and interpret the behavior and efficiency of iterative optimization heuristics. However, a limitation of this approach is its focus solely on individual module contributions, without emphasizing the interactions between modules and the exact values of modules used in the configuration. In our previous study [26], we used f-ANOVA to analyze the impact of individual modules and their interactions on the performance of modCMA-ES and modDE. The results, aggregated across 24 optimization problems, provided a general view of module importance but did not reveal how it varies across different problem types, which is the objective of this study. Unlike the earlier work, which focused only on interaction importance, the current study also examines the specific module values behind these interactions and their relation to high-problem landscape features.

In [28], f-ANOVA is introduced and used to quantify the importance of hyperparameters of ML models for a single dataset. In [36], f-ANOVA is used to quantify and contrast the hyperparameter importance of machine learning algorithms for a number of datasets and provide descriptive statistics across all datasets to infer their importance in general. In [43], f-ANOVA is used to assess the performance of hyperparameters of quantum neural networks. However, in none of the studies, the features of the datasets are considered, nor hyperparameter importance is used to obtain groups of datasets with similar hyperparameter importance.

## 3. Methodology for identifying module importance patterns across problem classes

This section describes our approach to identifying groups of problem classes where modules and their combinations have similar importance and contrasts module importance across groups. We start by generating problem class-specific datasets where modules serve as features of the modular algorithm variants, and the corresponding performance on the problem class is treated as the target in the dataset. The datasets capture the relationship between the algorithmic modules and the performance in a specific problem class. The approach consists of the following steps:

**(1) Quantifying individual and interaction module effects with f-ANOVA:** The problem class-specific datasets are used as an input to f-ANOVA [28], which decomposes the variance in the performance and systematically attributes it to the individual modules and their combinations. Assuming that we have $n$ modules that define the variants, the analysis results will quantify the effect (i.e., importance) on performance for $n$ individual modules, $\binom{n}{2}$ pairwise module interactions and $\binom{n}{3}$ triple module interactions. This is performed for each dataset. Higher-order interactions are computationally expensive to estimate. Further details will be provided later.

**(2) Grouping problem classes based on module effects:** To examine how the effects of modules vary across different problem classes, first, all quantified effects are concatenated into a vector representation for each problem class. The vector representation has size $n + \binom{n}{2} + \binom{n}{3} \times 1$. Having the representation for each problem class, a clustering algorithm is applied to explore which problem classes exhibit similarity in module effects and which are different in this aspect. This analysis is performed in high dimensions using the entire vector representation to prevent any information loss caused by data transformations to lower dimensions.

**(3) Comparison to the categorization of BBOB problem classes based on High-level features:** Clustering alignment scores are used to quantify how well the clusters from the previous step align with known high-level features of the problem classes, such as multi-modality or conditioning. This provides valuable insights into the extent to which the algorithm's module importance is influenced by these features and where discrepancies occur.

**(4) Algorithm configuration in the subspace of important modules:** To validate our analysis, we use the results to select a set of important modules and their best options—those associated with the best marginal performance. For each problem class, we then choose the corresponding algorithm variant from the data. We evaluate its performance by comparing it against several baseline algorithm selection strategies for new problem instances from the corresponding problem class.

## 4. Experimental design

In this section, we first describe the data, including the configuration space, the problem suite, and the algorithm performance data. Then, the datasets used by f-ANOVA are presented in more detail.

### 4.1. Configuration space

This study involves the modCMA-ES framework [24], a modular implementation of the CMA-ES algorithm. The framework provides a diverse choice of modules based on several state-of-the-art CMA-ES variants from where the design choices were translated into a set of modules, and allows for the creation of at least 36,288 modular variants. In this work, we reuse the performance data from [44] where six modules are considered. Table 2 lists the modules and briefly explains their functionality. More information about their practical module interactions, implementation details, and pseudocode can be found in the original Refs. Van Rijn et al. [45] and de Nobel et al. [24]. Table 3 shows the options taken into consideration for the modules, respectively. The different algorithm variants are created with a Cartesian product over the considered module options, resulting in 324 modCMA-ES variants.

### 4.2. Problem suite

The noiseless BBOB benchmark suite is considered in the experiments. It consists of 24 problem classes, where multiple instances of each problem class can be derived by using linear and non-linear transformation processes of the base problem class. These involve shifting, rotating the axes, and scaling the coordinates of each basic function. The COCO platform [53] already consists of multiple instances per problem class. In this study, we consider the first 5 instances of each of the 24 BBOB problem classes, in two problem dimensions $d = 5$ and $d = 30$, totaling 120 problem instances per problem dimension.

**Table 2**
List of the modules considered for the analysis of module importance, along with a brief description of their role in the modCMA-ES algorithm.

| module_id | Description |
|---|---|
| 1. base_sampler | In CMA-ES new generation of solutions is sampled from a multivariate normal distribution. Instead of random sampling from the distribution denoted as the *Gaussian* option, quasi-random sequences, such as *Sobol* or *Halton* sequences [46] can be used, to ensure an even coverage across the search space. |
| 2. elitist | The mechanism enables the best individual from the current population to be preserved into the next generation. When elitism is *On*, the best solution found so far is never lost. When elitism is *Off*, all solutions are subject to variation, and the best solution may be replaced. |
| 3. local_restart | When the optimization process stagnates, the algorithm is "restarted" from a different point in the search space. If set to *Off*, restarts are not performed. *IPOP* [47] increases the population size after every restart by a constant factor. *BIPOP* alternates between larger and smaller population sizes as the optimization process progresses [48]. |
| 4. mirrored_sampling [49] | A technique where each newly sampled solution has its mirrored counterpart (sign-reversed version) added to the generation. When mirroring is *Off*, the search process operates without generating mirrored solutions. With *mirrored pairwise*, only the best solution of each pair is used in the weighted recombination procedure. |
| 5. weights_option [50] | The mean of the multivariate normal distribution for the next generation is computed as a weighted average of solutions from the current generation, where weights determine each solution's influence on the calculation. *Equal* weights treat all solutions equally, while the *default* and $(1/2)\hat{}\lambda$ give progressively less influence to individuals based on their quality, with the $(1/2)\hat{}\lambda$ being more aggressive in declining the weights. |
| 6. step_size_adaptation [51] | This module is used to dynamically adjust parameters such as the step size or mutation rate throughout the optimization process. *PSR* (Population Success Ratio) [52] offers immediate feedback by adjusting step size based on the success of the current generation, making it highly responsive. In contrast, *CSA* (Cumulative Step Size Adaptation) uses a long-term trend analysis through the cumulative sum of successful steps to adjust step size gradually. |

**Table 3**
Considered options of the six modCMA modules (324 option combinations in total).

| module_id | option_id |
|---|---|
| 1. base_sampler | Gaussian[a], Sobol, Halton |
| 2. elitist | Off[a], On |
| 3. local_restart | Off[a], IPOP, BIPOP |
| 4. mirrored_sampling | Off[a], mirrored, mirrored pairwise |
| 5. weights_option | default[a], equal, $(1/2)\hat{}\lambda$ |
| 6. step_size_adaptation | csa[a], psr |

[a] The default option for each module is denoted.

### 4.3. Performance data

The repository in [44] contains the performance data of 324 modCMA-ES variants. Using the IOHexperimenter platform [54], each modCMA variant was run 10 independent times on the first five instances of each of the 24 problem classes in the BBOB suite on the COCO platform, in two problem dimensions $d \in \{5, 30\}$. The budget was set to $1500d$ function evaluations. The algorithm performance is measured as the distance of the best-obtained solution to the optimum (*distance*). The values are trimmed so that $10^{-9}$ is the lowest value, as this distance is sufficiently close to the optimum. We consider the median distance of the ten runs as the final performance of a variant on a problem instance. We then log-transform the values by using a logarithm with base 10. The *distance* metric has a lower bound of $-9$, where a lower distance indicates better performance.

### 4.4. Quantifying module importance with f-ANOVA

In this study, we aim at quantifying module importance for each of the 24 problem classes. The data is first partitioned into 24 datasets, based on the problem class. Since each problem includes five instances, with 324 configurations evaluated on them, we use the first four instances for the f-ANOVA experiment and reserve the fifth for validating its explanations in algorithm configuration analysis. Each dataset is represented by 324 configurations (data instances), characterized by the six modules (features), and linked to performance on the corresponding problem class. To ensure robustness, for each modCMA configuration, we take the median performance across the four instances used in the f-ANOVA experiment. The calculation of module effects for each problem class was the computationally most demanding part of the experiment. All experiments were performed on a computer equipped with an 11th Gen Intel(R) Core(TM) i7-1165G7 CPU, 16 GB of RAM, and two GPUs: an NVIDIA T500 with 4 GB of GDDR6 VRAM and Intel(R) Iris(R) Xe Graphics with 128 MB VRAM.

**Table 4**
Hyperparameter search space for the HC algorithm.

| Hyperparameter | Range |
|---|---|
| *n_clusters* | {2, ..., 24} |
| *metric* | {cosine, euclidean} |
| *linkage* | {single, complete, average} |

### 4.5. Clustering

The results from running f-ANOVA on each problem separately end up as 24 vector representations, each consisting of the module effects along with their pairwise and triple interactions. This means we obtained a dataset with 24 data instances, representing the BBOB problems, described by $6+\binom{6}{2}+\binom{6}{3}$ effects used as features for clustering. We employ Hierarchical Clustering (HC) [55] to automatically group the problem classes based on the module effects. Given the small size of the dataset (24 data instances) and the aspect of interpretability, we opted for the HC algorithm. This method provides a transparent view of the clustering process through a dendrogram, allowing us to trace how clusters merge at each step.

To determine the optimal number of clusters, we perform a grid search over the hyperparameters of the clustering algorithm and use the Silhouette coefficient [56] as a metric for clustering quality. The score ranges from $-1$ to 1, and a higher score indicates better separation of clusters, i.e., dense and well-separated clusters. The tested hyperparameters along with the ranges are listed in Table 4.

## 5. Results

The results section categorizes the findings into four subsections: analysis of the performance data, the grouping of problem classes based on module effects, correlating the groupings with the high-level landscape features, and finally, the algorithm configuration in the subspace of important modules, spanning two distinct problem dimensions.

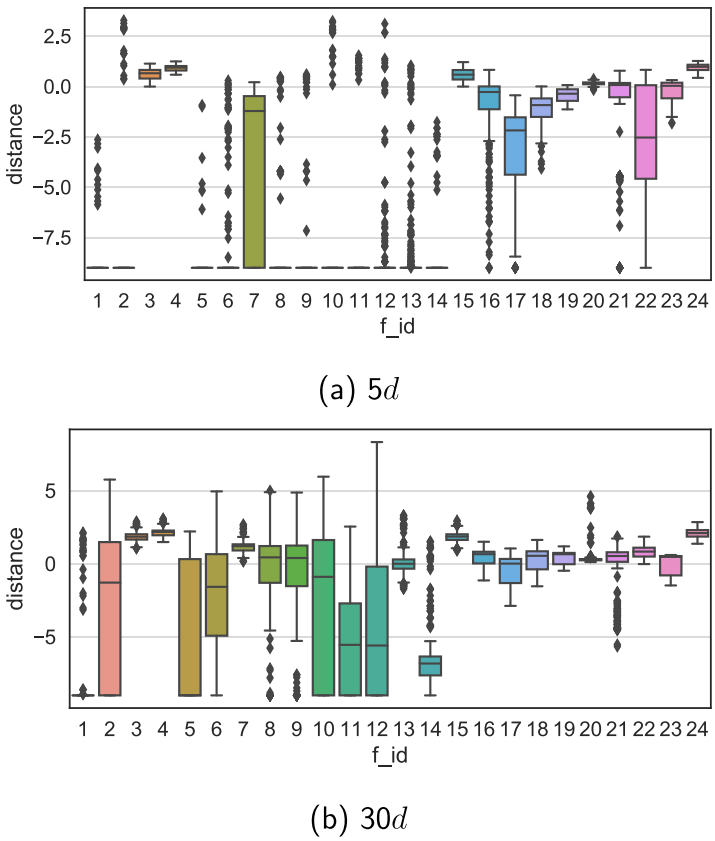



**Fig. 2.** The distribution of performance for 324 modCMA-ES variants across 24 problem classes, evaluated at two dimensions, $d \in 5, 30$. Performance is measured using the $distance$ metric, which quantifies the difference between the quality of the best-found solution (within a budget of $1500d$ function evaluations) and the global optimum. The values are presented on a logarithmic scale.

### 5.1. Algorithm performance distribution

This section analyzes the performance distribution of the 324 modCMA-ES variants on each of the 24 problem classes. The algorithm performance is measured using the $distance$ metric, defined as the difference between the quality of the best-obtained solution within a budget of $1500d$ solution evaluations and the global optimum. The raw distance values are then preprocessed by applying a log transformation and aggregation by applying the median as a more robust metric over multiple problem instances, as described in the experimental design, to obtain the final performance of a variant for a problem class.

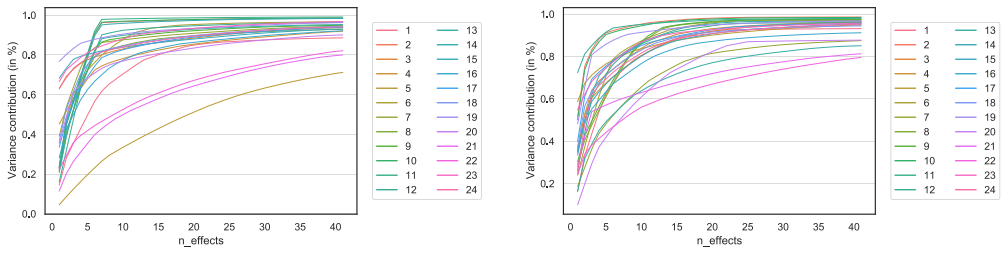
Fig. 2 presents the distribution of performance, for the 324 modCMA-ES variants on each of the 24 problem classes, spanning two problem dimensions $d \in \{5, 30\}$. The spread of the boxplots reflects the variability in the performance of different variants, allowing us to assess the potential for algorithm configuration. For low problem dimensionality of $d = 5$, there is very low variability in performance, with the majority of variants achieving the best possible effectiveness of $-9$. This suggests that for simpler problems, most configurations can solve them. However, the presence of some strong outliers indicates that a few variants still perform notably worse, emphasizing the importance of avoiding poor configurations. Certain problem classes—specifically those with $f_i d \in 7, 16, 17, 22$—exhibit greater performance variability even at $d = 5$, implying that these problems are more sensitive to algorithm configuration.

For high problem dimensionality of $d = 30$, the variability in performance increases in general. This suggests that as problem complexity grows, selecting the right algorithm configuration becomes increasingly crucial. In these cases, tuning the algorithm can lead to substantial performance improvements.

### 5.2. Cumulative module importance

Here we evaluate how much performance variance stems from the considered module effects and whether more complex interactions drive performance. Applying f-ANOVA to a dataset dedicated to a problem class results in quantifying the individual effect (of each of the six modules), the pairwise interaction effects (for each of the $\binom{6}{2} = 15$ module pairs), and the triplet interaction effects ($\binom{6}{3} = 20$ module triples). This totals 41 module effects. Each effect represents the proportion of variance, out of the total variance in the performance, that is contributed by a specific module or combination of modules. A higher variance contribution signifies the higher importance of the corresponding module(s). Summing all the considered effects reflects the cumulative variance contribution, ranging from 0 (no variance contribution) to 1 (the total variance in performance). Please note that we consider up to triplet module interactions, while higher-order interactions may also exist but are not explicitly analyzed in this study.

The cumulative variance contribution is visualized in Fig. 3. The $x$-axis lists the number of module effects (isolated + pairwise + triple, or how many effects out of 41), while the $y$-axis shows the cumulative variance contribution. The lines reflect how the variance contribution accumulates as more module effects are considered. In low problem dimensionality ($5d$), the steep slope at the beginning indicates that a large portion of the variance can be attributed to a small subset of module effects. In some cases, a single module effect contributes over 60% of the total variance in performance. The module effects collectively account for over 80% of the total variance in performance, in each of the 24 problem classes. The substantial proportion means that the algorithms' behavior is shaped by combinations of up to three modules, while higher-order interactions (involving four or more modules) have relatively little contribution. This underscores that the




(a) $5d$ (b) $30d$

**Fig. 3.** Number of effects vs. cumulative variance contribution of the 41 module effects, for each of the 24 problem classes in the BBOB suite. The subplots correspond to (a) $5d$ and (b) $30d$ problems.

conducted analysis offers a comprehensive insight into which modules or module combinations significantly influence the performance of the modCMA-ES algorithm.

There are problem classes with a more gradual slope and lower cumulative variance contribution. For the 5th problem class, this can be attributed to the extremely low variability in the performance of the variants, as seen in Fig. 2. Thus, f-ANOVA is unable to attribute variance to specific modules, as there is effectively no meaningful difference between the module options. For the problem classes with $f\_id \in \{21, 22\}$, although there is a high variability in performance (see Fig. 2), the cumulative variance remains low. This suggests that higher-order interactions — beyond triplet effects — contribute to the variance, which are not quantified by the current analysis.

In high problem dimensionality ($d = 30$), the module effects collectively account for over 80% of the variance in performance in each of the 24 problem classes. However, there is a more gradual increase in the cumulative variance contribution, thus a larger number of module effects is needed to reach the same amount of variance contribution as in the low dimensional case. This indicates that module interactions become more influential as problem complexity increases, emphasizing the importance of capturing not just individual but also higher-order effects when analyzing algorithm behavior in more challenging problem landscapes.

We need to mention here that the 80% threshold was mentioned only to illustrate that, when cumulatively summed within each problem class, the importance of all single modules, pairs, and triplet interactions generally account for over 80% of the explained performance variance. This threshold is not used for any analysis. This is intended as a general indication that those interactions contribute significantly to performance. The number of effects contributing to different cumulative percentages are also visualized in Fig. 3(a).

### 5.3. Grouping of problem classes based on module importance

We now present the results of grouping problem classes based on the module effects. For each problem class, a meta-representation of 41 values is created by combining the individual effects and interaction (pairwise and triple) effects of all six modules. The HC algorithm is applied to the meta-representations to cluster them automatically. The clustering is performed using the high-dimensional meta-representations, to prevent any information loss caused by transforming data to lower dimensions.

#### 5.3.1. 5d results

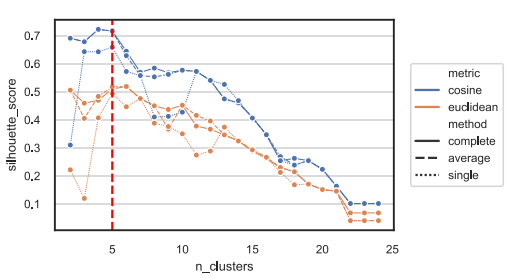
The clustering results for low problem dimensionality ($d = 5$) are displayed in Fig. 4 as a set of three sub-figures. The top left of Fig. 4 shows the outcomes from the tuning of the clustering algorithm for different numbers of clusters, distance metric, and linkage methods. The $x$-axis shows the number of clusters. The y-axis shows the Silhouette coefficient used to measure clustering quality, which ranges from −1 to 1, with higher values indicating better-defined clusters. The lines correspond to different hyperparameter option combinations. We set the optimal number of clusters to five, the metric to $cosine$, and the method to $average$, as indicated by the red vertical line. The average distance between the clusters of 0.3 (see Appendix A.1), and the high silhouette score indicate that the chosen clustering setup effectively groups problem classes. Fig. 4(b) presents a visualization of the clusters in 2D. The visualization is generated by applying Principal Component Analysis (PCA) [57], a dimensionality reduction technique that transforms the high-dimensional meta-representations into a lower-dimensional space. The $dim0$ and $dim1$ axes represent the new coordinates in the projected space. Each point on the plot represents a problem class ($f\_id \in \{1, \dots 24\}$). The color coding reflects the detected clusters.

Fig. 4(c) displays the clustering results in the form of a clustermap. The rows of the clustermap correspond to different problem classes denoted by $f\_id$, while the columns to the module effects denoted by $effect\_id$. The color coding reflects the variance contribution of the module effect, ranging from 0 (indicating no importance) to 1 (high importance). The first column corresponds to the cluster a problem class is assigned to, where each cluster is denoted by a different color. From the results we can see that most of the problem classes are divided into two larger clusters - the $red$ and $cyan$ clusters.

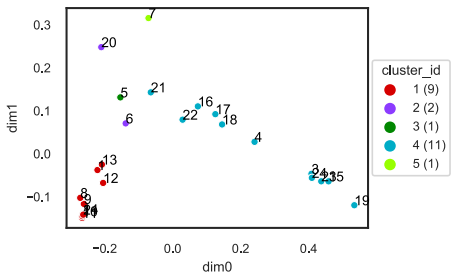
In the $red$ cluster ($f\_id \in \{1, 2, 8 - 14\}$), the high importance of the *elitist*, *mirrored*, and the *weights option* modules, becomes evident through their individual effect, as well as their pairwise and triplet interaction effects. The synergy of the three mentioned modules determines the behavior of the modCMA-ES algorithm on the uni-modal problem classes that characterize this cluster [29].

Fig. 5 displays the marginal performance of the options for the $mirrored$, $weights\ option$ and $elitist$ module and their pairwise combinations, on the 12th problem class from the BBOB suite. The problem class was selected randomly as the patterns are similar to the other problem classes in the cluster. The boxplots display the marginals for the individual effect, the $x$-axis shows the different options for the module, and the $y$-axis indicates the marginal performance, where lower values indicate better marginal performance (i.e., lower distance to the optimum on average). We can observe that turning "On" elitism results in better performance, indicating that focusing on refining good solutions is more effective than broader exploration in the absence of deceptive local optima. Next, both strategies of "Off" and "mirrored" show the same effectiveness, while "pairwise mirroring" is detrimental to the performance. It has been shown that mirrored sampling introduces a bias towards smaller step sizes [49], causing the new solutions to be very similar to the existing ones, which slows down convergence. This explains why in some cases turning mirroring "Off" can be beneficial. The $weights\ option$ module reveals that non-uniform weighting schemes, specifically the "default" and $(1/2)\hat{}\lambda$, are more effective, which is in line with the standard in evolutionary computation nowadays [49].

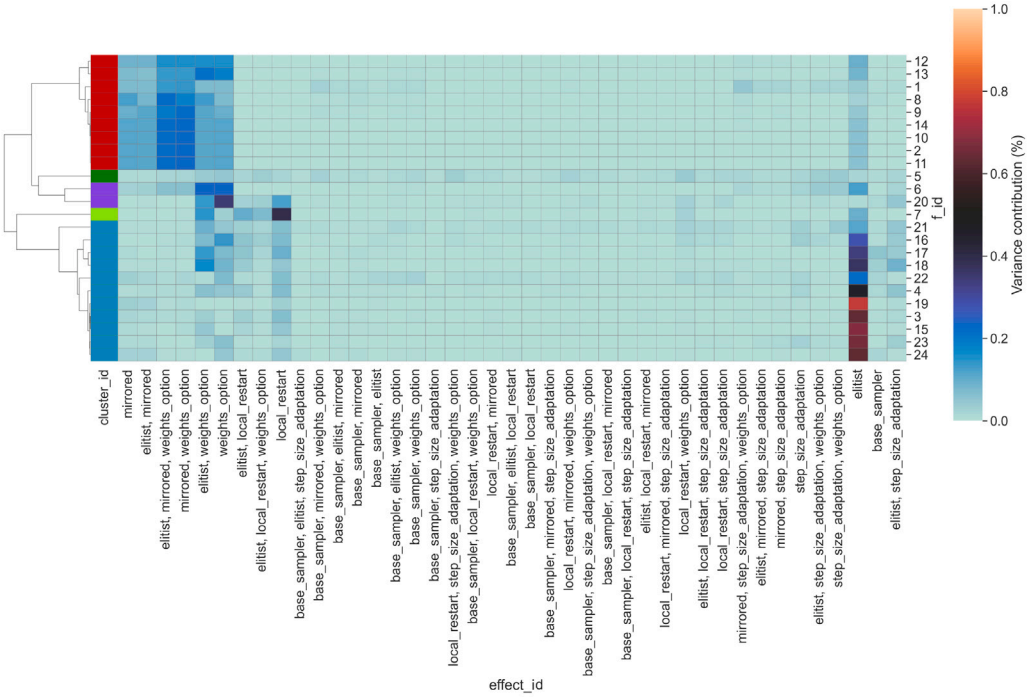
The three heatmaps (represented in Fig. 5) display the marginal performance for a combination of module options, the $x$ and $y$-axis show




(a) Performance results from the tuning of the HC algorithm. Silhouette coefficient is used as clustering quality measure.




(b) Visualization of the clusters of problem classes in 2D using PCA.




(c) Clustermap showing the clustering of the vector representations. The dendrogram illustrates how different problem classes cluster based on the module effects.

**Fig. 4.** Clustering results of the vector representations of problem classes for low problem dimensionality ($d = 5$).

the different options for the involved modules correspondingly, while the color-coding indicates the marginal performance, where lower values indicate better performance (i.e. lower distance to the optimum on average). We can see that enabling elitism consistently leads to better performance. However, when elitism is disabled, performance drops, particularly in the presence of pairwise mirroring. This suggests that when elitism is enabled to prioritize top solutions, mechanisms like mirroring have less impact. However, with disabling elitism the search process relies more on other mechanisms such as mirroring. Next, the pairing of elitism with the different weighting schemes shows that regardless of whether elitism is turned "On" or "Off", the non-uniform weighting schemes consistently result in better performance. Additionally, disabling elitism and assigning equal weights to all solutions significantly harms performance. This indicates that even if you do not explicitly enable elitism, giving the top solutions much higher weights can have the same effectiveness because it favors exploitation. Lastly, pairing mirroring and different weighting schemes shows that using non-uniform weighting schemes consistently yields better performance, regardless of the mirroring option. However, when using equal weights the performance is worse, especially in the pairwise mirroring case. This suggests that with prioritization of top performing solutions with non-uniform weighting schemes, mirroring has less impact, while disabling them shifts reliance to mechanisms like mirroring. In summary, this underscores how critical it is to configure the weighting strategy in uni-modal landscapes. The other two modules are also important, but their effectiveness largely depends on the weights option scheme.

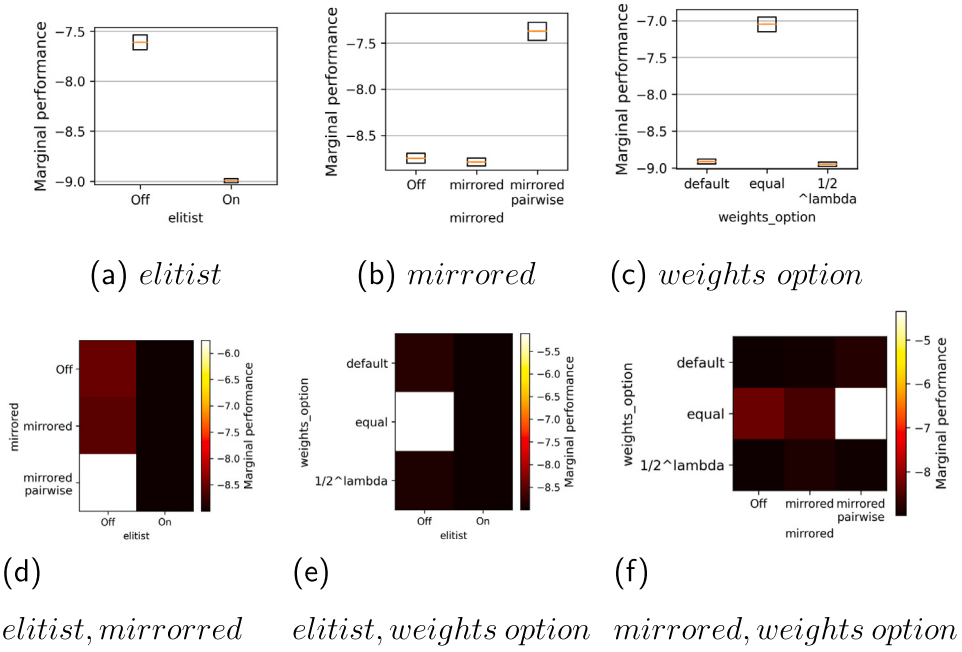



**Fig. 5.** Marginal performance of the modules on the 12th problem class from the BBOB suite, for low problem dimensionality. Marginal performance is the average performance of a specific option for a module, computed across all possible combinations of the other modules. The subplots correspond to different module effects indicated in the title of the subplot.

The *cyan* cluster ($f\_id \in \{3, 4, 15-19, 21-24\}$) is characterized by the strong influence of the *elitist* module, along with notable variance contributions from *local restart*, and *weights option*, both individually and in interaction with *elitist*. These three modules jointly shape the behavior of modCMA-ES on the multi-modal problems that define this cluster [29].

This aligns with established knowledge that enabling elitism on multi-modal problems can hinder exploration and cause premature convergence to local optima. Also, as shown in [58], integrating restart mechanisms improves CMA-ES performance on multi-modal problems. This is reflected in Fig. 6, which illustrates the marginal performance of the *elitist*, *local restart*, and *weights option* modules and their pairwise interactions on problem class 16. We can see that enabling elitism consistently leads to poorer performance, regardless of the restart strategy, suggesting that restarts alone cannot counteract its negative effects. Next, non-uniform weights prove highly effective when elitism is disabled, suggesting that weighted recombination is a reliable strategy for navigating multi-modal landscapes. In summary, configuring elitism is crucial in multi-modal landscapes, as it strongly influences the effectiveness of the other two modules.

Finally, the cluster map reveals some more isolated cases of module importance patterns, such as the large individual effect of the *local restart* module for the 7th problem class (Step Ellipsoidal). The problem class consists of many plateaus of different sizes, apart from a small area close to the global optimum, the gradient is almost zero everywhere, making it easy for algorithms to get stuck. Thus the very high importance of the *local restart* module is aligned with our intuition that it helps the algorithm escape these flat regions. The fact that the proposed analysis reveals this verifies that it functions as intended. Another isolated case is the 5th problem class (Linear Slope), for which the cluster map indicates that the importance attributed to the module effects is consistently low across all modules, leading to the lowest total variance observed of 70%. This occurs because the Linear Slope represents a simple, linear problem landscape, and nearly all variants of modCMA effectively solve this problem landscape. This makes the algorithm performance relatively insensitive to changes in the module settings, consequently, the module effects have minimal importance.

### 5.3.2. 30d results

The clustering results for high problem dimensionality ($d = 30$) are displayed in Fig. 7. Based on the clustering performance results in Fig. 7(a), we set the optimal number of clusters to five, the metric to *cosine*, and the method to *complete*, indicated by the red vertical line. The average distance between the clusters of 0.3 (see Appendix A.1), and the moderate silhouette score of 0.6 indicate that the chosen clustering setup effectively groups problem classes. On the top right, Fig. 7(b) presents a visualization of the clusters in 2D. The clustering suggests that the problem classes can be mainly divided into five groups, with one larger grouping, the *red* cluster, and a few smaller clusters.

The *red* cluster is characterized by the highly important *elitist* module, followed by the *base sampler* module. Its problem classes largely overlap with the *cyan* cluster in low problem dimensionality. The high importance of the *base sampler* stems from the instability of the Halton sampler in high dimensions, where its samples cluster, introducing bias and degrading performance [59,60]. This is visible in Fig. 8, which displays the marginals for the *elitist*, *base sampler* module, and their combination, for the 17th problem class. When considered individually, we can observe that for *base sampler*, both "Sobol" or "Gaussian" show as effective, while the marginal performance for the "Halton" option is significantly worse. When combined together, when elitism is enabled the performance is worse regardless of the option for *base sampler*. When elitism is "Off", pairing it with "Sobol" or "Gaussian" sampler can be very effective, while "Halton" remains the least effective option.

The exception to the multi-modal functions in the *red* cluster is the 7th problem class (Step Ellipsoid) becomes increasingly challenging for the algorithm variants as problem complexity rises, which can be seen in Fig. 2(b). The importance of local restarts diminishes, while the role of elitism grows, mirroring the module importance pattern on multi-modal problem classes.

Another case is the 5th problem class (Linear Slope). Unlike at $d = 5$, the *elitist* module becomes very important for that problem class. This suggests that the problem landscape becomes more complex in higher dimensions, making the performance sensitive to the module options. Although the problem class is uni-modal the module importance pattern resembles the one on multi-modal problem classes. Upon closer examination of the marginals of the module options, we observed that

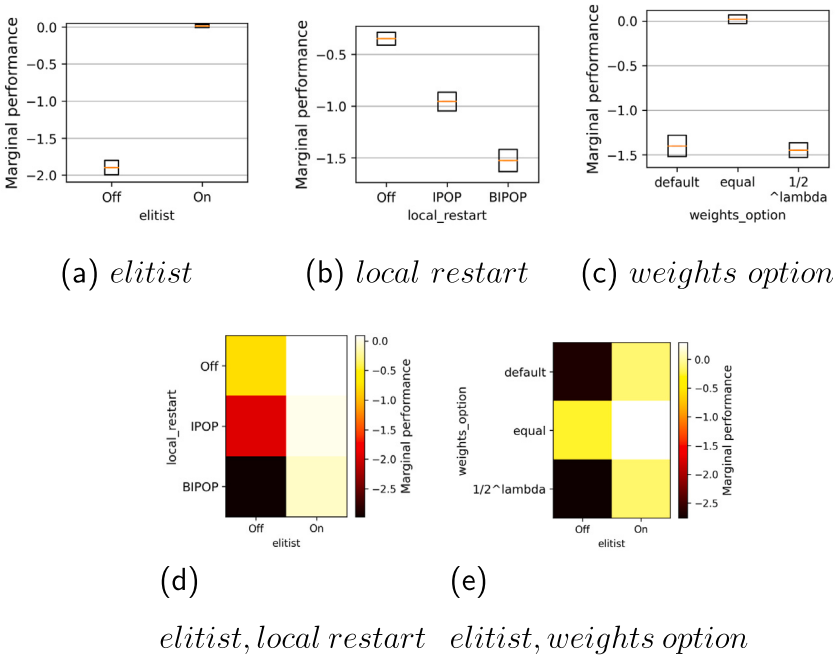



**Fig. 6.** Marginal performance of the modules on the 16th problem class from the BBOB suite, for low problem dimensionality. Marginal performance is the average performance of a specific option for a module, computed across all possible combinations of the other modules. The subplots correspond to different module effects indicated in the title of the subplot.

they differ from the rest of the cluster. Therefore, the modules can hold comparable importance across different problem classes but exert different optimal options (i.e., values). Fig. 9 features the marginals of the *elitist* module on the 5th problem class. We can observe that, unlike the 17th problem class, here the "On" option is the most effective.

In the *lightgreed* cluster ($f\_id \in \{2, 10-12\}$), the importance of the individual effect of the *base sampler* is even more pronounced, making it very influential for the performance of modCMA in high dimensions overall. The effect of the *elitist* module in this cluster is not as pronounced as in the *red* cluster, making an obvious difference between these two groups. To a lesser extent, the *weights options* and *step size adaptation* modules also play a role, through their individual effects but also interactions with elitist and base sampler modules. This cluster consists of uni-modal landscapes with high scaling.

Fig. 10 displays the marginals for the *base sampler*, *elitist*, *weights option* modules and their combination, for the 2nd problem class. We select this problem class as representative of the problem classes in this cluster. From the combination of elitism and the different weighting options, we can see that a careful configuration is needed. When elitism is "Off", non-uniform weighting schemes are more effective, whereas with elitism "On", equal weights tend to perform better.

### 5.4. Comparison to the categorization of BBOB problem classes based on high-level features

The clustering of the problem classes is evidently aligned with the multi-modality landscape feature. In this section, we present a more detailed analysis of how the other high-level landscape features correlate to the clusters.

To quantify the alignment between the clusters of the BBOB problem classes based on the module effects and the categorization based on the high-level features (predefined categories), we utilize homogeneity, completeness, and V-Measure as metrics. The metrics are chosen because they are independent of the absolute number of groups, i.e., it does not require the resulting number of groups to match from in both strategies. This is because they are normalized, making them robust to differences in the number of clusters between the two grouping results. Homogeneity measures whether each cluster contains only problem classes of a single predefined category; completeness measures if all problem classes from a predefined category are assigned to the same cluster. Their harmonic mean results in the V-measure. To prioritize the purity of the clusters, ensuring that each cluster is as internally consistent as possible with respect to the predefined categories, we compute the V-measure with a higher weight for homogeneity set at 0.6. All metrics are bounded between 0 and 1, where higher values indicate better alignment. We present the results separately for low and high problem dimensionality.

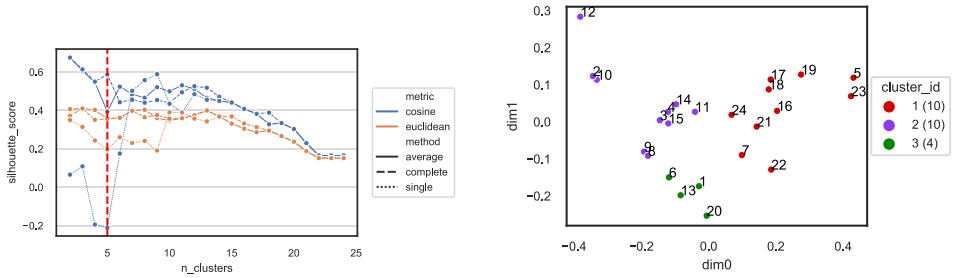
**Table 5**
Alignment scores between predefined high-level feature categories and the clusters based on module effects, for low problem dimensionality ($d = 5$).

| High-level feature | Homogeneity | Completeness | v-Measure |
|---|---|---|---|
| multim. | 0.557 | 0.558 | 0.557 |
| basins | 0.469 | 0.422 | 0.450 |
| gl.-loc. | 0.431 | 0.382 | 0.411 |
| gl.-struc. | 0.417 | 0.389 | 0.406 |
| scaling | 0.220 | 0.242 | 0.228 |
| funnel | 0.333 | 0.125 | 0.205 |
| separ. | 0.215 | 0.109 | 0.157 |
| homog. | 0.129 | 0.082 | 0.107 |

#### 5.4.1. 5d results

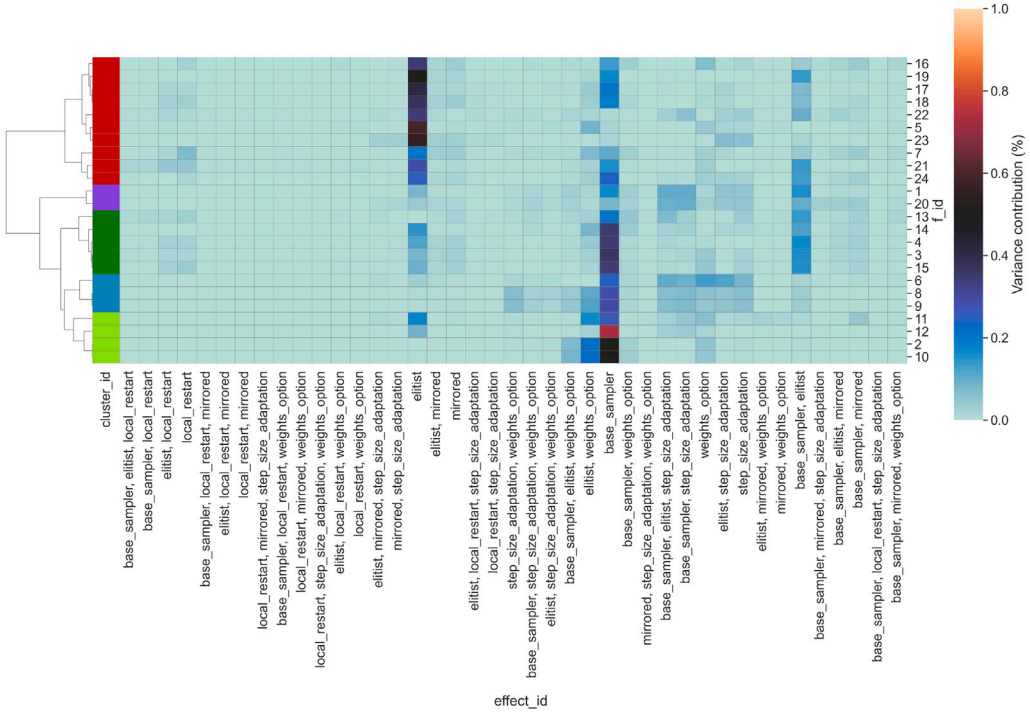
Table 5 presents the alignment scores for low problem dimensionality ($d = 5$). For each high-level feature, we calculate the alignment score between its predefined categories and the clusters. The alignment scores are ordered by the V-Measure in descending order. The results demonstrate that the clusters align most closely with the multi-modality categories, followed by the basins, and global-to-local categories. The lowest alignment scores are demonstrated for the separability and homogeneity categories. This implies that the module importance in modCMA is strongly influenced by whether a problem class is multi-modal or uni-modal, and less by whether a problem class is separable or not.

Fig. 11 illustrates the alignment of the clusters with the predefined categories of the multi-modality, basins, and global to local feature, in the form of a contingency matrix. The rows of the matrix correspond



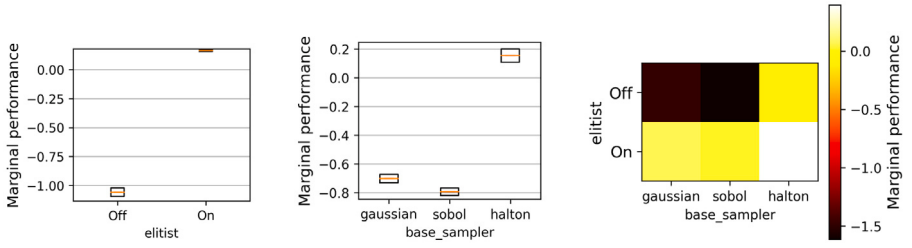

(a) Performance of the HC algorithm for different number of clusters.

(b) Visualization of the clusters in 2D using PCA.




(c) Cluster-map showing the module effects. The dendrogram illustrates how different problem landscapes cluster based on the importance of module effects.

**Fig. 7.** Clustering results of the vector-representations of problem classes for high problem dimensionality ($d = 30$).




(a) *elitist* (b) *base sampler* (c) *base sampler, elitist*

**Fig. 8.** Marginal performance of the modules on the 17th problem class in the BBOB suite, for high problem dimensionality. The subplots correspond to (a) *elitist* and (b) *base sampler* modules and (c) their combination.

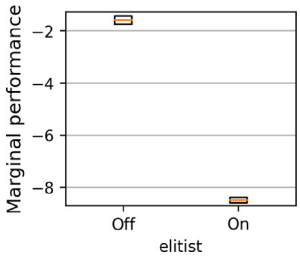



**Fig. 9.** Marginal performance of the *elitist* module on the 5th problem class from the BBOB suite, for high problem dimensionality.

**Table 6**
Alignment scores between the predefined high-level feature categories and the clusters based on module effects, of the BBOB problem classes, in high problem dimensionality ($d = 30$).

| High-level feature | Homogeneity | Completeness | v-Measure |
|---|---|---|---|
| scaling | 0.523 | 0.471 | 0.502 |
| multim. | 0.390 | 0.321 | 0.361 |
| gl.-struc. | 0.387 | 0.297 | 0.347 |
| homog. | 0.438 | 0.229 | 0.326 |
| basins | 0.270 | 0.200 | 0.239 |
| gl.-loc. | 0.260 | 0.189 | 0.228 |
| funnel | 0.378 | 0.117 | 0.205 |
| separ. | 0.040 | 0.016 | 0.026 |

to the predefined categories, while the columns represent the clusters. The cells in the matrix quantify how many problem classes from each predefined category are assigned to the corresponding cluster, with higher numbers indicating better alignment between the predefined categories and the clusters. In Fig. 11(a), we observe that the cluster with *cluster_id* of one primarily consists of seven uni-modal problem classes denoted by the *none* category, along with two problem classes with low multi-modality. On the other hand, the cluster with *cluster_id* of four is composed exclusively of multi-modal problem classes, with varying levels of multi-modality from low to high. Figs. 11(b) and 11(c) show that the cluster *cluster_id* of one is predominantly made up of problem classes without basins and no global-to-local contrast, while *cluster_id* of four contains problem classes mainly with basins and global-to-local contrast.

Although there is some alignment between the predefined categories and the clusters, it is not perfect. For instance, problem classes from the same predefined group can be assigned to different clusters, indicating that the module importance is different for them. For example, separability does not appear to be a particularly influential feature in determining the performance of modular CMA-ES. Regardless of separability, performance is more strongly driven by other landscape features. These findings also open several promising directions for future research and exploration. This suggests that the module importance in modCMA-ES is influenced by factors beyond just the high-level landscape features. The mixing of close categories like 'none' and 'low' in our analysis points to potential issues in how distinctions are drawn between these levels. These findings underscore the need for further investigation into the factors driving the algorithm behavior.

#### 5.4.2. 30d results

The results for high problem dimensionality ($d = 30$) are presented in Table 6. Here, the clusters align most closely with the scaling categories, followed by the multi-modality and global structure categories. The lowest alignment scores are demonstrated for the separability categories.

We can see that the features are different compared to the low dimensional case. This may be attributed to the emergent properties of the landscape as dimensionality increases, such as the increased interaction effects between variables, a larger number of local optima, etc., not captured by these predefined categories. This finding highlights the importance of developing more nuanced approaches to problem categorization that bridge the gap between landscape features and algorithm behavior (see Fig. 12).

### 5.5. Algorithm configuration in the subspace of important modules

To validate the findings regarding module importance derived from f-ANOVA, we conduct a simple algorithm configuration experiment. Without employing meta-learning or hyperparameter tuning techniques, we assess whether the insights obtained from f-ANOVA for each problem class can guide the selection of an effective configuration for solving a different instance within the same class. The validation instance is excluded from the f-ANOVA experiment to ensure an independent evaluation. While we acknowledge that this instance belongs to the same problem class, our goal is to determine how closely we can approximate the optimal configuration, thereby indirectly testing the practical value of the extracted knowledge.

To this end, we set a threshold for the contributed variance and identify the module effects that, together, cumulatively account for at least this specified amount. We select module effects sequentially, starting with the highest contributing effect and continuing in descending order until the threshold is met. We then configure the most important modules (appearing in the module effects that satisfied the threshold) using the option values that exhibit the best marginal performance, while keeping the remaining modules set to their default values. This is done for each problem class separately. We then select the corresponding variant from the 324 modCMA-ES variants available. Tables 7 and 8 display the number of module effects and the configured modules for each problem class, for the threshold of 80% set for the variance contribution in $d = 5$ and $d = 30$ correspondingly.

To evaluate the performance of the selected configurations based on f-ANOVA, we compare their performance against established baselines for algorithm configuration: the Virtual Best Solver (VBS): the best-performing configuration for each problem instance based on the raw performance results; the Single Best Solver (SBS): the configuration that performs best on average across all problem instances; the Default Configuration (DC): the configuration with default options for all modules. We compute the mean absolute error (MAE) and mean square error (MSE) of the baseline selectors (SBS and DC) and the f-ANOVA-based selector relative to the VBS across all problem classes. A lower error signifies that the selector is identifying configurations closer to the VBS. The VBS, SBS, and DC are shown in Appendix A.2.

We need to mention here that a direct comparison with existing automated configuration tools is an unfair benchmarking approach. Our evaluation is based on an exhaustive pool of all possible configurations derived from the six modCMA-ES modules, while tools like iRace or SMAC search only a subset of this space and ultimately return a single configuration. Consequently, we benchmark against the virtual best solver from our pool to estimate the configuration error, as this represents the best outcome that tools like iRace [61] or SMAC [62] could potentially achieve.

Table 9 presents the MAE and MSE of the baselines and the selector based on f-ANOVA with respect to the VBS, in $d = 5$. For f-ANOVA, the selector identifies the best module options for the important modules (*fanova_best_option*). However, we also select the worst option values to compare the difference with the VBS (*fanova_worst_option*), with the expectation to lead to poor performance results. Additionally, we conducted a sensitivity analysis on the variance contribution threshold, using 60%, and 80% to select the contributing modules. We also tested 100% by configuring all six modules to their best option value. However, it is important to note that achieving 100% variance contribution is not possible, as we only consider individual, pairwise, and triple interactions. We can observe that the configuration based on fanova_best

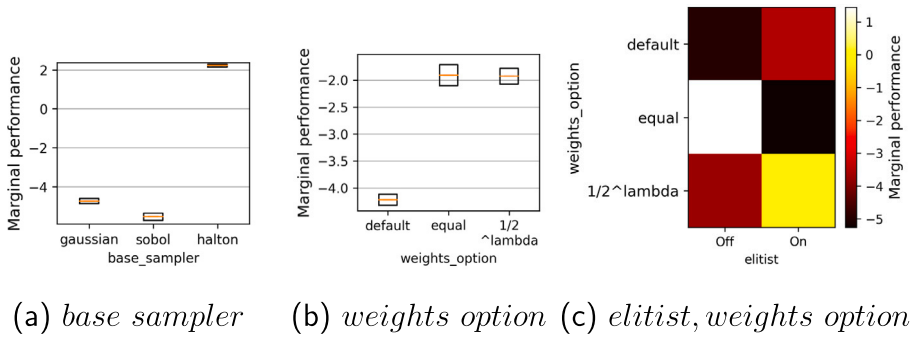



**Fig. 10.** Marginal performance of the modules on the problem class with *f_id* of two in the BBOB suite, for low problem dimensionality. Marginal performance is the average performance of a specific option for a module, computed across all possible combinations of the other modules. The subplots correspond to different module effects indicated in the title of the subplot.

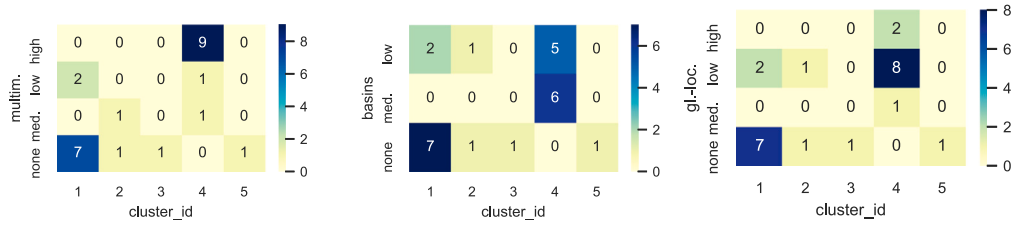



**Fig. 11.** Contingency matrix of the high-level features and clusters, for low problem dimensionality ($d = 5$).

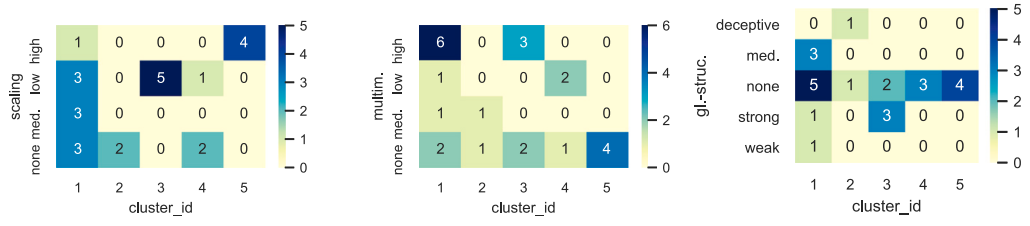



**Fig. 12.** Contingency matrix of the high-level features and clusters, for high problem dimensionality ($d = 30$).

with 80% threshold performs very close to the results achieved by the SBS, and better than the default configuration that was expected. This indicates that based on f-ANOVA good configurations can be selected. The advantage of the fanova_best selector is that we know what module of the modCMA algorithm is the most important for the problem instances, while from SBS does not provide this information. Also, we can see that fanova_worst selector performs worse than all the selectors, indicating that using the insights from f-ANOVA bad configurations can be recognized. This approach guides the selection of optimal module options while avoiding suboptimal ones. These insights serve as a strong starting point for advanced algorithm configuration tools (e.g., iRace [61], SMAC [62]), highlighting which modules should be tuned rather than optimizing the entire module space.

Table 10 shows the error of the selectors with respect to the VBS, in $d = 30$. Here, the selector based on f-ANOVA performs better than the SBS and the default selector. These results stem from the fact that, in $d = 5$, nearly all configurations exhibit similar performance, regardless of their specific settings, due to the low variability in the raw performance data. In contrast, for $d = 30$, all thresholds outperform the SBS due to the greater diversity in performance outcomes. We emphasize that the obtained results confirm the usefulness of f-ANOVA insights in understanding the significance of modules for specific landscapes.

### 5.6. Algorithm configuration for unseen problem classes

To assess the generalization of the findings regarding module importance, we conducted an algorithm configuration experiment where findings from the most similar known problem class based on the eight high-level features guide the configuration on an unseen class. In this setup, for the test problem class, we identified the most similar class among the remaining 23 using the eight features, using cosine similarity. Then we configured the most important modules for the most similar class. The resulting configuration is then evaluated on the test problem class and compared to the best-known configurations on it (i.e., VBS from the ground truth performance data) and the SBS, which is the configurations that performs the best across all problem instances from the 23 problems of the train data. In low dimensions, the results in Table 11 show that the fanova_best selector yields performance close to the single best solver (SBS) and outperforms the default, while fanova_worst performs the worst. In high dimensions, from Table 12, fanova_best achieves the lowest error relative to the virtual best solver, surpassing both default and SBS configurations. This demonstrates the potential to transfer modCMA-ES configurations to unseen problem classes.

**Table 7**
Selected modules for each of the 24 BBOB problem classes in $d = 5$.

| f_id | Number of module effects | module_id | Number of modules |
|---|---|---|---|
| 1 | 6 | elitist, mirrored, weights_option | 3 |
| 2 | 3 | elitist, mirrored, weights_option | 3 |
| 3 | 1 | elitist | 1 |
| 4 | 3 | elitist, local_restart, weights_option | 3 |
| 5 | 26 | base_sampler, elitist, local_restart, mirrored, step_size_adaptation, weights_option | 6 |
| 6 | 2 | elitist, weights_option | 2 |
| 7 | 2 | elitist, local_restart, weights_option | 3 |
| 8 | 3 | elitist, mirrored, weights_option | 3 |
| 9 | 3 | elitist, mirrored, weights_option | 3 |
| 10 | 3 | elitist, mirrored, weights_option | 3 |
| 11 | 3 | elitist, mirrored, weights_option | 3 |
| 12 | 3 | elitist, mirrored, weights_option | 3 |
| 13 | 3 | elitist, mirrored, weights_option | 3 |
| 14 | 3 | elitist, mirrored, weights_option | 3 |
| 15 | 1 | elitist | 1 |
| 16 | 4 | elitist, local_restart, weights_option | 3 |
| 17 | 3 | elitist, local_restart, weights_option | 3 |
| 18 | 2 | elitist, weights_option | 2 |
| 19 | 1 | elitist | 1 |
| 20 | 2 | elitist, weights_option | 2 |
| 21 | 16 | base_sampler, elitist, local_restart, mirrored, step_size_adaptation, weights_option | 6 |
| 22 | 14 | base_sampler, elitist, local_restart, mirrored, step_size_adaptation, weights_option | 6 |
| 23 | 1 | elitist | 1 |
| 24 | 1 | elitist | 1 |

**Table 8**
Selected modules for each of the 24 BBOB problem classes in $d = 30$.

| f_id | Number of module effects | module_id | Number of modules |
|---|---|---|---|
| 1 | 4 | base_sampler, elitist, step_size_adaptation | 3 |
| 2 | 1 | base_sampler | 1 |
| 3 | 2 | base_sampler, elitist | 2 |
| 4 | 2 | base_sampler, elitist | 2 |
| 5 | 1 | elitist | 1 |
| 6 | 3 | base_sampler, elitist, step_size_adaptation, weights_option | 4 |
| 7 | 8 | base_sampler, elitist, local_restart, mirrored, weights_option | 5 |
| 8 | 4 | base_sampler, elitist, step_size_adaptation, weights_option | 4 |
| 9 | 4 | base_sampler, elitist, step_size_adaptation, weights_option | 4 |
| 10 | 1 | base_sampler | 1 |
| 11 | 2 | base_sampler, elitist | 2 |
| 12 | 1 | base_sampler | 1 |
| 13 | 8 | base_sampler, elitist, local_restart, step_size_adaptation, weights_option | 5 |
| 14 | 2 | base_sampler, elitist | 2 |
| 15 | 3 | base_sampler, elitist | 2 |
| 16 | 4 | base_sampler, elitist, mirrored, weights_option | 4 |
| 17 | 1 | elitist | 1 |
| 18 | 2 | base_sampler, elitist | 2 |
| 19 | 1 | elitist | 1 |
| 20 | 9 | base_sampler, elitist, mirrored, step_size_adaptation | 4 |
| 21 | 7 | base_sampler, elitist, local_restart, mirrored, step_size_adaptation | 5 |
| 22 | 12 | base_sampler, elitist, mirrored, step_size_adaptation, weights_option | 5 |
| 23 | 1 | elitist | 1 |
| 24 | 2 | base_sampler, elitist | 2 |

**Table 9**
Comparison of performance for the different selectors using MAE, MSE, in $d = 5$.

| Threshold | selector_id | Mae | Mse |
|---|---|---|---|
| | default | 2.23 | 16.44 |
| | sbs | 0.44 | 1.02 |
| 0.6 | fanova_best_option | 1.42 | 7.28 |
| | fanova_worst_option | 5.63 | 50.99 |
| 0.8 | fanova_best_option | *0.86* | 4.76 |
| | fanova_worst_option | 6.43 | 58.12 |
| 1.0 | fanova_best_option | 0.87 | 5.19 |
| | fanova_worst_option | 6.42 | 58.41 |

**Table 10**
Comparison of performance for the different selectors using MAE, MSE, in $d = 30$.

| Threshold | selector_id | Mae | Mse |
|---|---|---|---|
| | default | 1.08 | 5.51 |
| | sbs | 0.69 | 1.37 |
| 0.6 | fanova_best_option | **0.47** | 0.61 |
| | fanova_worst_option | 6.46 | 71.49 |
| 0.8 | fanova_best_option | ***0.36*** | 0.47 |
| | fanova_worst_option | 7.17 | 82.91 |
| 1.0 | fanova_best_option | **0.43** | 0.72 |
| | fanova_worst_option | 7.01 | 78.01 |

## 6. Discussion

Our primary objective was to uncover and interpret interaction patterns among algorithmic modules within modCMA-ES, which we subsequently validated within the context of algorithm configuration. This perspective fills an important gap in understanding how and why specific combinations of algorithmic modules succeed or fail across different problem landscapes. We firmly believe that our empirical

**Table 11**
Comparison of performance for the different selectors using MAE in $d = 5$.

| Threshold | selector_id | Mae |
|---|---|---|
| | default | 2.226658 |
| | sbs | 1.134190 |
| 0.6 | fanova best | 1.626263 |
| | fanova worst | 5.335198 |
| 0.8 | fanova best | 1.458332 |
| | fanova worst | 5.354431 |
| 1.0 | fanova best | 1.445394 |
| | fanova worst | 5.396609 |

**Table 12**
Comparison of performance for the different selectors using MAE in $d = 30$.

| Threshold | selector_id | Mae |
|---|---|---|
| | default | 1.084986 |
| | sbs | 1.403190 |
| 0.6 | fanova best | 0.931137 |
| | fanova worst | 6.285828 |
| 0.8 | fanova best | 0.457566 |
| | fanova worst | 6.589164 |
| 1.0 | fanova best | 0.489965 |
| | fanova worst | 6.284557 |

findings can inspire a range of new theoretical research directions. Importantly, we are not proposing a new algorithm configuration method. Rather, our aim was to evaluate whether the derived explanations can effectively guide the configuration process toward solutions with performance comparable to state-of-the-art configurations. The experimental results demonstrate that these explanations offer highly promising guidance. From a practical standpoint, such insights can be integrated into automated configuration tools (e.g., iRace or SMAC) to restrict the search space to the most influential modules. Furthermore, they can serve as external knowledge inputs for LLM-based systems [63] tailored to recommend optimal configurations for specific optimization problems.

We would like to emphasize that our approach — using f-ANOVA to analyze module interactions — is general and can be readily extended to larger module spaces or other modular optimization algorithms, such as modular differential evolution (modDE) [64] or component-based particle-swarm optimization framework (PSO-X) [65]. The primary goal of our analysis is to understand the behavior of algorithmic frameworks, rather than introducing new modules or operators without a clear understanding of their actual benefits — an issue that often contributes to the "elephant in the room" in current algorithm development [8].

One challenge of f-ANOVA is its assumption of independence among modules — an assumption we have also adopted in our analysis. However, the observed importance of certain module interactions — and, more importantly, the specific values at which they occur — suggests promising directions for future research into potential dependencies between modules. Another challenge when applying f-ANOVA is the low variability in performance, as it depends on differences among variants to identify important options. When most variants perform similarly, detecting key options becomes difficult. However, rather than being a limitation, this also suggests that algorithm configuration may not be necessary in such cases.

Another challenge is that the analysis depends on a full enumeration of the configuration space, making it computationally demanding. To address this, sharing data across studies within structured resources, such as the OPTION ontology [66], is essential. This approach not only reduces the computational cost of testing new modules but also promotes data reuse, enabling the evaluation of new methods and direct comparisons with other experiments.

While high-level landscape features like multi-modality and separability aid interpretation, they provide only a limited view of the problem landscape in relation to algorithm behavior. Our methodology overcomes this limitation by adopting a dynamic, data-driven approach to categorize problem landscapes based on algorithm-module interactions. This enables a more precise representation, particularly in high-dimensional settings where traditional methods may fail to capture critical features.

Regarding algorithm configuration, we primarily validate the usefulness of f-ANOVA insights in guiding module selection. The results have been promising, demonstrating that in the future these insights can help hyperparameter optimization tools by narrowing the search space to focus on the most critical modules, thereby improving efficiency and reducing computational costs.

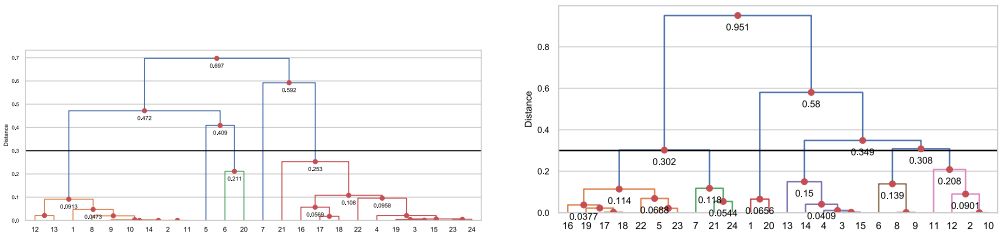
One limitation is that f-ANOVA insights have only been validated within individual problem classes, limiting their generalization to unseen problem classes not included in the experiments. In the future, we aim to leverage knowledge from the 24 BBOB problem classes. There is potential to transfer insights using high-level features, though identifying them for new problems remains a challenge. Thus, if a new problem class emerges, we can first represent it using low-level problem landscape features such as exploratory landscape analysis [67], topological landscape analysis [68], DeepELA [69], and TransOpt [70]. A similarity analysis can then identify the closest matching problem class from the BBOB dataset to test whether these insights can be transferred.

Furthermore, the module importance vector representations can be tested alongside problem landscape features to enhance performance prediction models. This could be particularly valuable for automated algorithm selection, enabling more accurate predictions by incorporating knowledge of which modules are most crucial for specific problem classes.

## 7. Conclusions

While the continuous expansion of the modular Covariance Matrix Adaptation Evolution Strategy (modCMA-ES) with new modules holds promise for performance gain, it introduces the need to systematically quantify the effect of individual modules and their interactions with the existing ones, in a large-scale benchmarking setting. Additionally, the performance of these modCMA-ES variants varies widely across different problems, indicating that the effects may vary depending on the landscape features of the optimization problem at hand. To address this we perform a problem-specific analysis of module importance using f-ANOVA to analyze the performance data from 324 modCMA-ES variants, on each of the 24 problem classes from the Black Box Optimization Benchmark (BBOB) suite. We extend the f-ANOVA analysis by using data-driven clustering of problem classes based on module importance and correlate these clusters with the high-level landscape features of the problem classes. Finally, to validate the findings on module importance we perform algorithm configuration by configuring the most important modules for new problem instances within the same problem class. The analysis is performed twice: once for low-dimensional problem instances with five dimensions and again for high-dimensional problems with thirty dimensions, to compare how module importance may vary not only with problem landscape features but also with problem complexity.

The tailored analysis shows that for low-dimensional problems, clustering indicates that the behavior of the modCMA-ES algorithm can be distinguished by two primary module importance patterns, corresponding to uni-modal and multi-modal landscapes. While previous evidence suggests the significance of these modules, this study is the first to empirically validate their interactions and importance across a diverse range of modCMA-ES variants, comparing their influence across different problem landscape types. Although the elitist and mirrored sampling modules are important, their effectiveness is highly influenced by the configuration of the weight option scheme. The key




(a) $5d$ (b) $30d$

**Fig. 13.** Dendrogram illustrating the hierarchical clustering process. The black horizontal line denotes the average distance between points from different clusters. The subplots correspond to (a) $5d$ and (b) $30d$ problems.

**Table 13**
The VBS, SBS, and the default configurator on the validation problem instances in $d = 5$.

| configurator_id | configuration_id | f_id | base_sampler | elitist | local_restart | mirrored | step_size_adaptation | weights_option |
|---|---|---|---|---|---|---|---|---|
| VBS | conf_0 | 1 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 2 | gaussian | On | Off | Off | csa | default |
| | conf_307 | 3 | halton | Off | Off | mirrored pairwise | psr | default |
| | conf_176 | 4 | gaussian | Off | IPOP | Off | csa | 1/2*lambda |
| | conf_0 | 5 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 6 | gaussian | On | Off | Off | csa | default |
| | conf_100 | 7 | halton | On | BIPOP | mirrored | csa | equal |
| | conf_0 | 8 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 9 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 10 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 11 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 12 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 13 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 14 | gaussian | On | Off | Off | csa | default |
| | conf_230 | 15 | gaussian | Off | IPOP | mirrored | csa | 1/2*lambda |
| | conf_202 | 16 | halton | Off | BIPOP | Off | csa | default |
| | conf_184 | 17 | sobol | Off | BIPOP | Off | csa | default |
| | conf_322 | 18 | halton | Off | BIPOP | mirrored pairwise | csa | 1/2*lambda |
| | conf_44 | 19 | halton | On | IPOP | Off | csa | equal |
| | conf_157 | 20 | halton | On | Off | mirrored pairwise | psr | 1/2*lambda |
| | conf_100 | 21 | halton | On | BIPOP | mirrored | csa | equal |
| | conf_15 | 22 | gaussian | On | IPOP | Off | psr | 1/2*lambda |
| | conf_252 | 23 | halton | Off | Off | mirrored | csa | default |
| | conf_226 | 24 | gaussian | Off | BIPOP | mirrored | csa | equal |
| SBS | conf_196 | | sobol | Off | BIPOP | Off | csa | 1/2*lambda |
| default | conf_162 | | gaussian | Off | Off | Off | csa | default |

factor determining modCMA-ES performance on multi-modal problems is the elitism module's configuration. While local restarts and weight options also play a significant role, their impact is largely dependent on how elitism is set up.

In high-dimensional problems for multi-modal problems, elitism continues to be the most important module, along with the base sampler module. In uni-modal problems, the base sampler plays a crucial role in performance, alongside the pairwise interaction between elitism and the weight option. The module effect patterns indicate a clear distinction between uni-modal and multi-modal problem classes in modCMA for low-dimensional problems. Notably, the grouping of problem classes remains largely consistent across both low and high dimensions, suggesting that multi-modality is the dominant factor influencing the algorithm's behavior.

The high-level feature analysis showed that the clusters align most strongly with multi-modality categories and least with separability categories, in low problem dimensionality. This suggests that module importance in modCMA-ES is primarily determined by whether a problem class is multi-modal or uni-modal. In high problem dimensionality the module importance is influenced by scalability, multi-modality, and global structure.

The verification step confirmed that configuring the key modules identified by f-ANOVA leads to improved performance over the default configuration, yielding results comparable to or better than the best-performing single variant in both low and high-dimensional problems.

## CRediT authorship contribution statement

**Ana Nikolikj:** Writing – original draft, Visualization, Methodology, Investigation, Formal analysis, Conceptualization. **Tome Eftimov:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

## Data and code availability

The data and code is available at the Zenodo repository [71].

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

We acknowledge the support of the Slovenian Research and Innovation Agency through program grant P2-0098, and project grants No. J2-4460 and No. GC-0001, and young researcher grant No. PR-12897 to AN. This work is also funded by the European Union under Grant Agreement 101187010 (HE ERA Chair AutoLearn-SI).

**Table 14**
The VBS, SBS, and the default configurator on the validation problem instances in $d = 30$.

| configurator_id | configuration_id | f_id | base_sampler | elitist | local_restart | mirrored | step_size_adaptation | weights_option |
|---|---|---|---|---|---|---|---|---|
| VBS | conf_0 | 1 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 2 | gaussian | On | Off | Off | csa | default |
| | conf_299 | 3 | sobol | Off | BIPOP | mirrored pairwise | psr | equal |
| | conf_220 | 4 | gaussian | Off | BIPOP | mirrored | csa | default |
| | conf_0 | 5 | gaussian | On | Off | Off | csa | default |
| | conf_162 | 6 | gaussian | Off | Off | Off | csa | default |
| | conf_252 | 7 | halton | Off | Off | mirrored | csa | default |
| | conf_162 | 8 | gaussian | Off | Off | Off | csa | default |
| | conf_166 | 9 | gaussian | Off | BIPOP | Off | csa | default |
| | conf_0 | 10 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 11 | gaussian | On | Off | Off | csa | default |
| | conf_0 | 12 | gaussian | On | Off | Off | csa | default |
| | conf_115 | 13 | gaussian | On | Off | mirrored pairwise | psr | equal |
| | conf_27 | 14 | sobol | On | IPOP | Off | psr | equal |
| | conf_221 | 15 | gaussian | Off | BIPOP | mirrored | psr | default |
| | conf_264 | 16 | halton | Off | Off | mirrored | csa | 1/2^lambda |
| | conf_296 | 17 | sobol | Off | IPOP | mirrored pairwise | csa | equal |
| | conf_239 | 18 | sobol | Off | BIPOP | mirrored | psr | default |
| | conf_240 | 19 | sobol | Off | Off | mirrored | csa | equal |
| | conf_205 | 20 | halton | Off | Off | Off | psr | equal |
| | conf_101 | 21 | halton | On | BIPOP | mirrored | psr | equal |
| | conf_163 | 22 | gaussian | Off | Off | Off | psr | default |
| | conf_275 | 23 | gaussian | Off | BIPOP | mirrored pairwise | psr | default |
| | conf_257 | 24 | halton | Off | BIPOP | mirrored | psr | default |
| SBS | conf_194 | | sobol | Off | IPOP | Off | csa | 1/2^lambda |
| default | conf_162 | | gaussian | Off | Off | Off | csa | default |

## Appendix

### A.1. Grouping of problem classes based on module effects

Fig. 13 presents visualization of the clustering process through a dendrogram. The dendrogram is a tree-like diagram where each leaf represents an individual data point, and branches indicate how points or clusters are iteratively merged based on their pairwise distances. The height at which two branches join corresponds to the distance between clusters, offering a clear representation of their relative similarity. You can "cut" the dendrogram at a certain height to form clusters. For the clustering, the cosine distance metric and average linkage were used to measure the dissimilarity between points and determine how clusters are formed. The height of the branches represents the average distance between data points in the respective clusters. The dendrogram was cut at a height of 0.3, allowing for a clear division of the data into five clusters in low problem dimensions (see Fig. 13a) and three clusters in high problem dimensions (see Fig. 13b).

### A.2. Algorithm configuration in the subspace of important modules

Tables 13 and 14 presents the VBS, SBS, and the default configurator on the validation problem instances in $d = 5$ and $d = 30$, respectively.

## Data availability

I will share a link to the data and code after article publication.

## References


[1] C. García-Martínez, P.D. Gutiérrez, D. Molina, M. Lozano, F. Herrera, Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness, Soft Comput. 21 (2017) 5573–5583.

[2] Y. Li, W. Gong, Multiobjective multitask optimization with multiple knowledge types and transfer adaptation, IEEE Trans. Evol. Comput. (2024).

[3] T. Zhang, D. Li, Y. Li, W. Gong, Constrained multitasking optimization via co-evolution and domain adaptation, Swarm Evol. Comput. 87 (2024) 101570.

[4] Y. Li, W. Gong, S. Li, Multitask evolution strategy with knowledge-guided external sampling, IEEE Trans. Evol. Comput. (2023).

[5] Y. Wang, C. Hu, F. Ming, Y. Li, W. Gong, L. Gao, A diversity-enhanced tri-stage framework for constrained multi-objective optimization, IEEE Trans. Evol. Comput. (2024).

[6] N. Hansen, A. Auger, D. Brockhoff, Data from the BBOB workshops, 2020.

[7] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, in: Proceedings of IEEE International Conference on Evolutionary Computation, IEEE, 1996, pp. 312–317.

[8] C. Aranha, C.L. Camacho Villalón, F. Campelo, M. Dorigo, R. Ruiz, M. Sevaux, K. Sörensen, T. Stützle, Metaphor-based metaheuristics, a call for action: the elephant in the room, Swarm Intell. 16 (1) (2022) 1–6.

[9] C.L. Camacho Villalón, T. Stützle, M. Dorigo, Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty, in: International Conference on Swarm Intelligence, Springer, 2020, pp. 121–133.

[10] C. Villalón, T. Stützle, M. Dorigo, Cuckoo search≡($\mu$+ $\lambda$)–evolution strategy, IRIDIA–Technical Report Series, 2021.

[11] U. Škvorc, T. Eftimov, P. Korošec, CEC real-parameter optimization competitions: Progress from 2013 to 2018, in: 2019 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2019, pp. 3126–3133.

[12] U. Škvorc, T. Eftimov, P. Korošec, GECCO black-box optimization competitions: progress from 2009 to 2018, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 275–276.

[13] C.L. Camacho-Villalón, M. Dorigo, T. Stützle, The intelligent water drops algorithm: why it cannot be considered a novel algorithm: A brief discussion on the use of metaphors in optimization, Swarm Intell. 13 (2019) 173–192.

[14] C.L. Camacho-Villalón, M. Dorigo, T. Stützle, Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors, Int. Trans. Oper. Res. 30 (6) (2023) 2945–2971.

[15] F. Campelo, C. Aranha, Sharks, zombies and volleyball: Lessons from the evolutionary computation bestiary, in: LIFELIKE Computing Systems Workshop 2021, CEUR-WS. org, 2021.

[16] M.A. Muñoz, H. Soleimani, S. Kandanaarachchi, Benchmarking algorithm portfolio construction methods, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2022, pp. 499–502.

[17] A. Kostovska, G. Cenikj, D. Vermetten, A. Jankovic, A. Nikolikj, U. Skvorc, P. Korosec, C. Doerr, T. Eftimov, PS-AAS: Portfolio selection for automated algorithm selection in Black-Box optimization, in: International Conference on Automated Machine Learning, PMLR, 2023, 1–17.

[18] G. Cenikj, R.D. Lang, A.P. Engelbrecht, C. Doerr, P. Korošec, T. Eftimov, Selector: selecting a representative benchmark suite for reproducible statistical comparison, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2022, pp. 620–629.

[19] R. Biedrzycki, Comparison with State-of-the-Art: Traps and pitfalls, in: 2021 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2021, pp. 863–870.

[20] N.G. Hall, M.E. Posner, The generation of experimental data for computational testing in optimization, in: Experimental Methods for the Analysis of Optimization Algorithms, Springer, 2010, pp. 73–101.

[21] M.A. Muñoz, K.A. Smith-Miles, Performance analysis of continuous black-box optimization algorithms via footprints in instance space, Evol. Comput. 25 (4) (2017) 529–554.

[22] A. Nikolikj, M.A. Munoz, T. Eftimov, Benchmarking footprints of continuous black-box optimization algorithms: Explainable insights into algorithm success and failure, Swarm Evol. Comput. 94 (2025) 101895.

[23] J.N. Hooker, Needed: An empirical science of algorithms, Oper. Res. 42 (2) (1994) 201–212.

[24] J. de Nobel, D. Vermetten, H. Wang, C. Doerr, T. Bäck, Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2021, pp. 1375–1384.

[25] N. van Stein, D. Vermetten, A. V. Kononova, T. Bäck, Explainable benchmarking for iterative optimization heuristics, ACM Trans. Evol. Learn..

[26] A. Nikolikj, A. Kostovska, D. Vermetten, C. Doerr, T. Eftimov, Quantifying individual and joint module impact in modular optimization frameworks, in: 2024 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2024, pp. 1–8.

[27] I. Soboĺ, Sensitivity estimates for nonlinear mathematical models, Math. Model. Comput. Exp. 1 (1993) 407.

[28] F. Hutter, H. Hoos, K. Leyton-Brown, An efficient approach for assessing hyperparameter importance, in: International Conference on Machine Learning, PMLR, 2014, pp. 754–762.

[29] N. Hansen, S. Finck, R. Ros, A. Auger, Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions, Technical Report RR-6829, INRIA, 2009, URL https://hal.inria.fr/inria-00362633/document.

[30] S. Cahon, N. Melab, E.-G. Talbi, Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics, J. Heuristics 10 (3) (2004) 357–380.

[31] N. Hansen, A. Auger, S. Finck, R. Ros, Real-parameter black-box optimization benchmarking 2009: Experimental setup, 2009.

[32] O. Mersmann, M. Preuss, H. Trautmann, Benchmarking evolutionary algorithms: Towards exploratory landscape analysis, in: Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, KrakÓw, Poland, September 11-15, 2010, Proceedings, Part I 11, Springer, 2010, pp. 73–82.

[33] P. Kerschke, M. Preuss, S. Wessing, H. Trautmann, Detecting funnel structures by means of exploratory landscape analysis, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, 2015, pp. 265–272.

[34] F. Hutter, H.H. Hoos, K. Leyton-Brown, Identifying key algorithm parameters and instance features using forward selection, in: Learning and Intelligent Optimization: 7th International Conference, LION 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers 7, Springer, 2013, pp. 364–381.

[35] N. Siegmund, A. Grebhahn, S. Apel, C. Kästner, Performance-influence models for highly configurable systems, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015, pp. 284–294.

[36] J.N. Van Rijn, F. Hutter, Hyperparameter importance across datasets, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2367–2376.

[37] A. Biedenkapp, M. Lindauer, K. Eggensperger, F. Hutter, C. Fawcett, H. Hoos, Efficient parameter importance analysis via ablation with surrogates, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 31, 2017, pp. 773–779.

[38] S. van Rijn, H. Wang, B. van Stein, T. Bäck, Algorithm configuration data mining for CMA evolution strategies, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2017, pp. 737–744.

[39] R.P. Prager, H. Trautmann, H. Wang, T.H. Bäck, P. Kerschke, 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, 2020, pp. 996–1003.

[40] A. Kostovska, D. Vermetten, S. Džeroski, C. Doerr, P. Korosec, T. Eftimov, The importance of landscape features for performance prediction of modular CMA-ES variants, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2022, pp. 648–656.

[41] A. Kostovska, D. Vermetten, P. Korošec, S. Džeroski, C. Doerr, T. Eftimov, Unveiling the Role of Modules: Assessing Importance and Classifying Modules in modCMA-ES and modDE via Algorithmic Behavior Analysis, 2023.

[42] A. Kostovska, D. Vermetten, P. Korošec, S. Džeroski, C. Doerr, T. Eftimov, Using machine learning methods to assess module performance contribution in modular optimization frameworks, Evol. Comput. (2024) 1–28.

[43] C. Moussa, Y.J. Patel, V. Dunjko, T. Bäck, J.N. van Rijn, Hyperparameter importance and optimization of quantum neural networks across small datasets, Mach. Learn. 113 (4) (2024) 1941–1966.

[44] A. Kostovska, D. Vermetten, S. Džeroski, P. Panov, T. Eftimov, C. Doerr, Using knowledge graphs for performance prediction of modular optimization algorithms, in: International Conference on the Applications of Evolutionary Computation (Part of EvoStar), Springer, 2023, pp. 253–268.

[45] S. Van Rijn, C. Doerr, T. Bäck, Towards an adaptive CMA-ES configurator, in: International Conference on Parallel Problem Solving from Nature, Springer, 2018, pp. 54–65.

[46] A. Auger, M. Jebalia, O. Teytaud, Algorithms (X, sigma, eta): Quasi-random mutations for evolution strategies, in: Artificial Evolution: 7th International Conference, Evolution Artificielle, EA 2005, Lille, France, October 26-28, 2005, Revised Selected Papers 7, Springer, 2006, pp. 296–307.

[47] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: 2005 IEEE Congress on Evolutionary Computation, Vol. 2, IEEE, 2005, pp. 1769–1776.

[48] N. Hansen, 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, 2009, pp. 2389–2396.

[49] A. Auger, D. Brockhoff, N. Hansen, Mirrored sampling in evolution strategies with weighted recombination, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, 2011, pp. 861–868.

[50] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159–195.

[51] O. Krause, T. Glasmachers, C. Igel, Qualitative and quantitative assessment of step size adaptation rules, in: Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, 2017, pp. 139–148.

[52] I. Loshchilov, A computationally efficient limited memory CMA-ES for large scale optimization, in: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, 2014, pp. 397–404.

[53] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, D. Brockhoff, COCO: A platform for comparing continuous optimizers in a black-box setting, Optim. Methods Softw. 36 (2020) 114–144, http://dx.doi.org/10.1080/10556788.2020.1808977.

[54] J. de Nobel, F. Ye, D. Vermetten, H. Wang, C. Doerr, T. Bäck, IOHexperimenter: Benchmarking platform for iterative optimization heuristics, 2021, CoRR abs/2111.04077 arXiv:2111.04077 URL https://arxiv.org/abs/2111.04077.

[55] D. Müllner, Modern hierarchical, agglomerative clustering algorithms, 2011, ArXiv abs/1109.2378 URL https://api.semanticscholar.org/CorpusID:8490224.

[56] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.

[57] I.T. Jolliffe, J. Cadima, Principal component analysis: a review and recent developments, Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. 374 (2065) (2016) 20150202.

[58] I. Loshchilov, M. Schoenauer, M. Sebag, Alternative restart strategies for CMA-ES, in: International Conference on Parallel Problem Solving from Nature, Springer, 2012, pp. 296–305.

[59] H. Faure, C. Lemieux, Generalized Halton sequences in 2008: A comparative study, ACM Trans. Model. Comput. Simul. (TOMACS) 19 (4) (2009) 1–31.

[60] J. de Nobel, D. Vermetten, T.H. Bäck, A.V. Kononova, Sampling in CMA-ES: Low numbers of low discrepancy points, 2024, arXiv preprint arXiv:2409.15941.

[61] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58.

[62] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, F. Hutter, SMAC3: A versatile Bayesian optimization package for hyperparameter optimization, J. Mach. Learn. Res. 23 (54) (2022) 1–9.

[63] N. van Stein, T. Bäck, Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics, IEEE Trans. Evol. Comput. (2024).

[64] D. Vermetten, F. Caraffini, A.V. Kononova, T. Bäck, Modular differential evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2023, pp. 864–872.

[65] C.L. Camacho-Villalón, M. Dorigo, T. Stützle, PSO-X: A component-based framework for the automatic design of particle swarm optimization algorithms, IEEE Trans. Evol. Comput. 26 (3) (2021) 402–416.

[66] A. Kostovska, D. Vermetten, C. Doerr, S. Džeroski, P. Panov, T. Eftimov, OPTION: Optimization algorithm benchmarking ontology, IEEE Trans. Evol. Comput. 27 (6) (2022) 1618–1632.

[67] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, G. Rudolph, Exploratory landscape analysis, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, 2011, pp. 829–836.

[68] G. Petelin, G. Cenikj, T. Eftimov, Tinytla: Topological landscape analysis for optimization problem classification in a limited sample setting, Swarm Evol. Comput. 84 (2024) 101448.

[69] M.V. Seiler, P. Kerschke, H. Trautmann, Deep-ela: Deep exploratory landscape analysis with self-supervised pretrained transformers for single-and multi-objective continuous optimization problems, Evol. Comput. (2025) 1–27.

[70] G. Cenikj, G. Petelin, T. Eftimov, A cross-benchmark examination of feature-based algorithm selector generalization in single-objective numerical optimization, Swarm Evol. Comput. 87 (2024) 101534.

[71] A. Nikolikj, Exploring module interactions in modular CMA-ES across problem classes, 2025, URL https://doi.org/10.5281/zenodo.15754445.