This is the Author-Accepted Version of the paper:

A. Nikolikj, M. A. Muñoz, E. Tuba and T. Eftimov, "Tracing the Interactions of Modular CMA-ES Configurations Across Problem Landscapes," *2025 IEEE Congress on Evolutionary Computation (CEC)*, Hangzhou, China, 2025, pp. 1-8, doi: 10.1109/CEC65147.2025.11042974.

© 2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

# Tracing the Interactions of Modular CMA-ES Configurations Across Problem Landscapes

Ana Nikolikj
*Computer Systems Department*
*Jožef Stefan Institute*
*Ljubljana, Slovenia*
*ana.nikolikj@ijs.si*

Mario Andrés Muñoz
*Computing and Information Systems*
*The University of Melbourne*
*Parkville, VIC 3010 Australia*
*munoz.m@unimelb.edu.au*

Eva Tuba
*Trinity University*
*San Anotnio, TX*
*Singidunum University*
*Belgrade, Serbia*
*etuba@ieee.org*

Tome Eftimov
*Computer Systems Department*
*Jožef Stefan Institute*
*Ljubljana, Slovenia*
*tome.eftimov@ijs.si*

***Abstract*—This paper leverages the recently introduced concept of algorithm footprints to investigate the interplay between algorithm configurations and problem characteristics. Performance footprints are calculated for six modular variants of the CMA-ES algorithm (modCMA), evaluated on 24 benchmark problems from the BBOB suite, across two-dimensional settings: 5-dimensional and 30-dimensional. These footprints provide insights into why different configurations of the same algorithm exhibit varying performance and identify the problem features influencing these outcomes. Our analysis uncovers shared behavioral patterns across configurations due to common interactions with problem properties, as well as distinct behaviors on the same problem driven by differing problem features. The results demonstrate the effectiveness of algorithm footprints in enhancing interpretability and guiding configuration choices.**

***Index Terms*—single-objective continuous optimization, landscape analysis, algorithm configuration footprint**

## 1. Introduction

Black-box optimization (BBO) involves developing and analyzing algorithms to tackle problems where the objective function is not explicitly accessible, relying only on iterative sampling and evaluation of candidate solutions [1]. In single-objective optimization (SOO), the aim is to identify the optimal solution from a set of candidates for a specific problem instance. A problem instance is defined by its variables outlining the solution space, an objective function for assessing solution quality, and a similarity measure between solutions, known as the neighborhood. Continuous optimization focuses on cases where these variables are real-valued, and the neighborhood is considered Euclidean. The process entails using an optimization algorithm to generate candidate solutions, which are evaluated by the objective function until convergence. These algorithms often introduce randomness by sampling from probability distributions over the problem space. Meta-heuristics, widely used in such contexts, offer resilient solutions even with limited computational resources or incomplete information [2]. They are typically divided into population-based and single-solution-based approaches.

Various algorithms have been proposed for continuous SOO, and their effectiveness is often assessed through statistical analyses, reporting average performance across a set of benchmark problems [2]. The algorithmic landscape encompasses various families, including covariance matrix adaptation evolution strategy (CMA-ES) [3], differential evolution (DE) [4], and particle swarm optimization (PSO) [5]. For example, each class of evolutionary algorithms incorporates specific mechanisms like mutation, crossover, selection strategies, and configurable hyperparameters. Algorithm instances representing different configurations of a given algorithm class may perform differently depending on the problem. Automated configuration techniques [6], [7] and hyper-heuristics [8] are employed to identify the best-performing configurations (the optimal combination of components or hyperparameters or both). These methods explore the relationship between algorithm performance and a predefined set of training problems, enabling dynamic selection and fine-tuning of algorithms.

In this direction, rather than analyzing algorithm configurations in isolation, the approach proposed in [9] advocates using a standardized modular optimization framework. This framework enables systematic evaluation of different algorithm configurations while ensuring consistent implementation across all variants. The core concept involves breaking down an algorithm into smaller, independent units known as modules. Each module offers configurable options that influence the algorithm's behavior and can be combined to form new algorithm variants. The studies highlight the benefits of modular frameworks [10]–[12], including more equitable implementation-based comparisons and the ability to explore interactions between different modules.

Although techniques from machine learning (ML), such as functional ANOVA (fANOVA and its extensions) [13] and Shapley values [14] from game theory, have been utilized to explain the influence of hyperparameters or modules by examining their impact on performance, notable challenges remain [15], [16]. Moreover, ablation studies were conducted to assess the impact of individual modules on performance across different problems, without exploring their properties in depth [17]. These explainable approaches have not fully bridged the gap in understanding the interaction between hyperparameters or modules and the properties of optimization problems. A recent approach was introduced to visually explore the relationships between algorithm performance, parameter settings, and characteristics of the problem landscape [18]. This entails jointly analyzing a 2D instance space, where problem instances are projected, and a 2D configuration space, where algorithm configurations are mapped. Building on the dimensionality reduction approach used in Instance Space Analysis, an optimization problem has been formulated to derive projections for these spaces, ensuring an interpretable relationship between them. This highlights the ongoing need for deeper insights into the interplay between algorithm configurations and problem properties.

**Our contribution**: We utilize a recently introduced approach known as algorithm footprints [19], [20] to explore the relationships between algorithm configurations and problem characteristics. Specifically, we generate algorithm configuration footprints for six modular CMA-ES (modCMA) configurations [10], tested on 24 problems from the BBOB benchmark suite [21], with separate evaluations in 5 and 30 dimensions. These footprints help explain why different configurations of the same algorithm yield varying performance outcomes and point out the problem characteristics driving those outcomes for each problem. Furthermore, the analysis highlights similar behavior across different configurations caused by shared interactions with problem properties and distinct behavior on the same problem due to the influence of different problem characteristics.

Section 2 provides an overview of relevant studies. Section 3 briefly explains the benchmarking footprints methodology. Section 4 outlines the experimental setup, while Section 5 presents the key results. Section 6 provides an interpretation of the findings and highlights future directions. Lastly, Section 7 concludes the paper.

## 2. Related work

Early efforts in this area focused on a detailed evaluation of module significance in CMA-ES by analyzing the performance contributions of individual modules through an enumeration of all possible module options [17]. However, this approach becomes impractical as the number of modules grows.

In [9] and [11] made use of the irace algorithm-tuning framework [6] to systematically survey a vast configuration space and uncover elite configurations—that is, the top-performing algorithm variants across a suite of optimization problems. Module importance was quantified based on the frequency of each module's inclusion in these elite configurations. This iterative process involved progressively expanding the module space explored by irace [6]. As additional modules were added, their interplay became more complex—an effect made visible through frequency-based plots of the elite configurations. Yet, a detailed quantitative breakdown that teases apart each module's contributions - whether individual, pairwise, or higher-order - has not yet been performed.

Recent research has investigated how different modules interact and influence configuration performance for modular CMA and modular Differential Evolution (modDE) using techniques like functional ANOVA and Shapley values [15], [16]. These approaches advance the analysis by measuring the impact of modules individually and their combinations (e.g. pairs and triplets) on overall performance. However, these analyses are limited to the algorithm performance space and fail to account for the characteristics of the given problems.

## 3. Algorithm configuration footprints

Workflow of the recently proposed benchmarking algorithm footprints methodology [20], [22] includes:

**(1) Training a meta-model**: A multi-target regression (MTR) model [23] is trained to predict algorithm performance based on the landscape features of the training problem instances. The MTR model leverages the capacity to predict multiple algorithm outcomes from shared features.

**(2) Meta-Representation Generation**: Apply explainability methods (e.g., SHAP) to calculate local feature importance for test instances, generating meta-representations that connect landscape features to algorithm performance. Each problem instance, paired with a specific algorithm, will have a unique meta-representation reflecting the relationship between its landscape features and its performance.

**(3) Meta-Representation Clustering**: Cluster meta-representations to identify regions with varying algorithm performance caused by different problem landscape features interactions. Rank clusters by their average performance, from lowest to highest.

**(4) Footprint Comparison**: Compare the distribution of meta-representations across clusters for different algorithms on the same instance to uncover similarities, differences, and challenging regions for the portfolio.

**(5) Feature Analysis**: Identify critical landscape features influencing problem difficulty.

In this study, we applied the methodology to a portfolio of algorithms from the same class but with different configurations. By tracing their interactions with problem landscapes, this approach helps explain how variations in configurations (hyperparameters or modules) of a core algorithm lead to differing performance outcomes.

## 4. Experimental design

**Performance data**: We leveraged on performance results from the earlier study by [24], which evaluated 324 modular CMA-ES configurations on the BBOB benchmark suite, running five instances per problem at dimensions $d = \{5, 30\}$. We selected six configurations per dimension (Table 1 for $5d$), including the best and worst performers (based on average performance) along with four standard CMA-ES variants, including elitist and local restart configurations. Performance was measured using the fixed-budget metric, assessing solution quality after $1500d$ function evaluations.

**Problem suite**: We employed 24 noiseless, single-objective black-box optimization problems from the BBOB benchmark suite [21]. Each problem has multiple instances generated through scaling, shifting, and rotation transformations. For this study, we used the first five instances of each problem for $d = \{5, 30\}$, resulting in two suites of 120 instances each.

**Landscape features**: Exploratory Landscape Analysis (ELA) feature data is reused from previous studies [24], [25]. The total of 46 features were computed using Sobol sampling with a sample size of $100d$ across 100 independent repetitions, and the median feature values were used to represent each problem instance. We applied forward feature selection to identify the most relevant features among the 46 available.

**MTR models**: We evaluated three machine-learning approaches to select a high-performing MTR model:Multi-Task Elastic Net (MTEN) [26], Random Forest (RF) [27], and Neural Network (NN) [28]. RF and MTEN were implemented using *scikit-learn* [29], while NN utilized the *keras* package [30].Before training, we applied min–max normalization to the ELA features, mapping them into the $[0, 1]$ interval. The normalization parameters were determined from the training set and then reused to transform the test set, ensuring consistency.

**MTR hyper-parameter tuning**: We performed hyperparameter optimization using Optuna's Python implementation [31] of the Tree-structured Parzen Estimator (TPE) sampler to pinpoint the best configuration. The process involves a budget of 50 trials: 40 trials utilize Random Search to explore the search space broadly, followed by 10 trials where TPE focuses on the most promising hyperparameter regions. The configuration whose hyperparameters maximize the cross-validation score on the training data is adopted as the optimal setup.

**Model evaluation**: The dataset comprises 120 instances - five instances of each of the 24 BBOB problem classes. One instance from each class (24 in total) is set aside as the test set, leaving 96 instances for training. We then carry out 4-fold cross-validation on the training data, stratifying folds by problem class. Each model is trained and validated across the four folds, and the average performance determines the best model. That model is finally retrained on all 96 training instances and evaluated against the 24 held-out test instances.

Model performance is assessed using Mean Absolute Error (MAE) and R-squared ($R^2$). MAE measures the average magnitude of the errors between predictions and actual values, while $R^2$ quantifies the fraction of variance in the true outcomes that the model explains, indicating their degree of alignment.

**SHAP explanations as meta-features:** Using the SHAP library [32], we compute problem instance-level feature contributions from the selected ML model, yielding an $n$-dimensional SHAP meta-feature vector

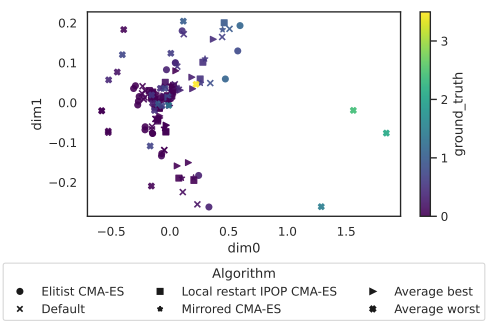
TABLE 1: The hyperparameter settings for the six modCMA configurations in $5d$.

| Name | elitist | mirrored | base_sampler | weights_option | local_restart |
|---|---|---|---|---|---|
| Default | False | Off | gaussian | default | Off |
| Elitism | True | Off | gaussian | default | Off |
| Mirrored sampling | False | mirrored | gaussian | default | Off |
| Local restart | False | Off | gaussian | default | IPOP |
| Best on average | False | mirrored | Sobol | default | BIPOP |
| Worst on average | False | pairwise | Halton | equal | Off |

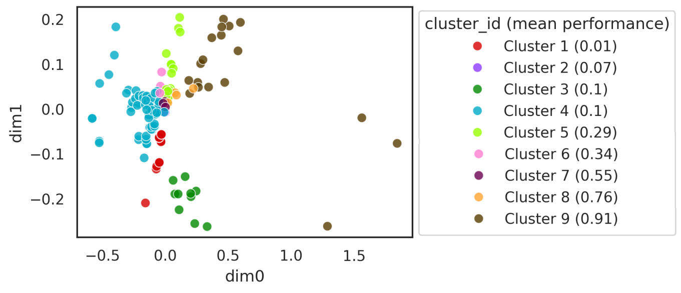
$$\mathbf{s} = (s_1, s_2, \ldots, s_n),$$

where each $s_i$ denotes the importance of feature $i$ to the algorithm performance.

**Clustering:** Hierarchical clustering [33] was applied to the meta-representations of problem instances to form groups automatically. Cluster quality was evaluated via the Silhouette coefficient—higher values correspond to more well-defined clusters—and we tuned hyperparameters by comparing Euclidean and cosine distance metrics, ultimately choosing the configuration that maximized the Silhouette score.



(a)



(b)

Figure 1: The problem instances represented by the SHAP meta-features are projected into a two-dimensional vector space, with point colors showing: (a) the true performance across the six modCMA-ES configurations; and (b) the cluster labels assigned to the five-dimensional ($5d$) test instances.

## 5. Results

Figures 1–3 present the footprint analyses for six modCMA configurations on $5d$ problems. In particular, Figure 1 illustrates (a) the ground-truth performance and (b) the resulting clusters of the problem SHAP meta-features.

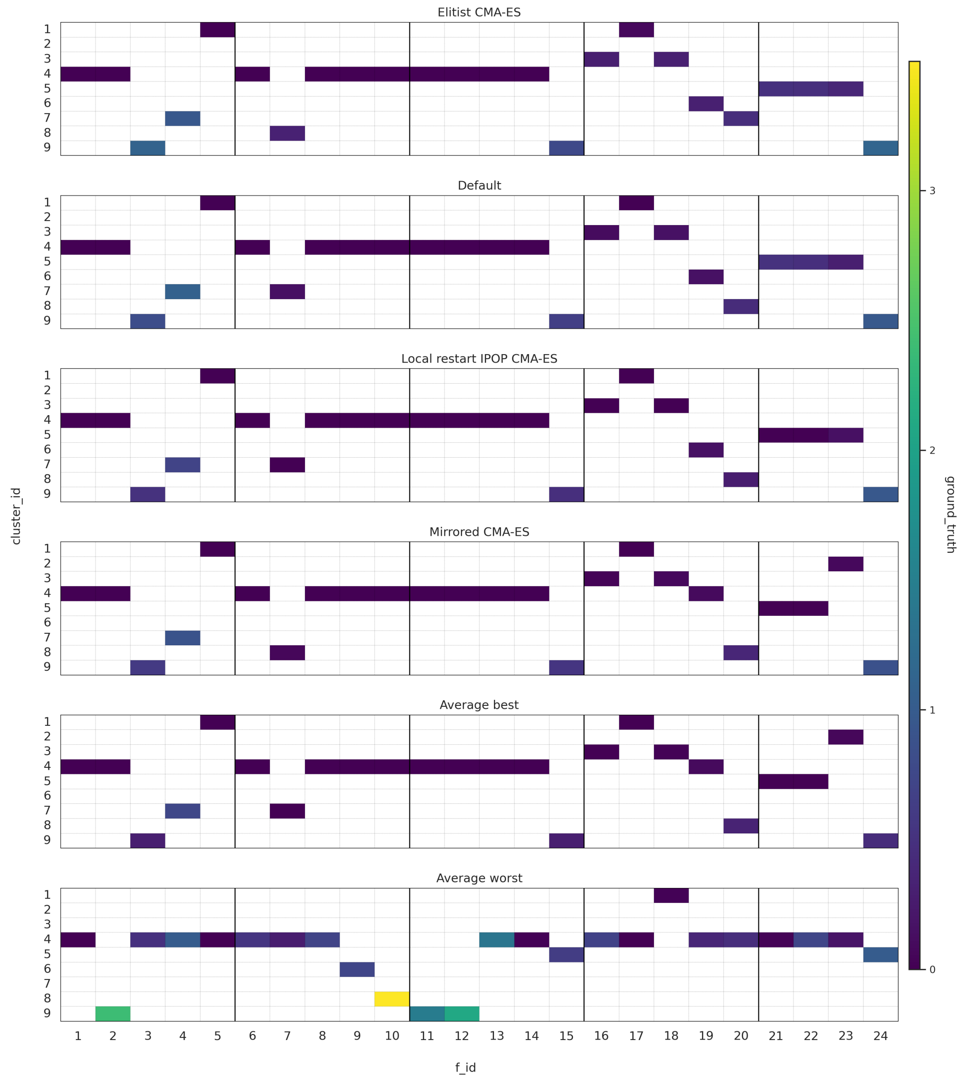
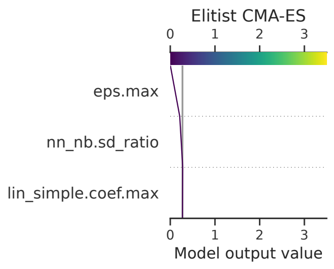


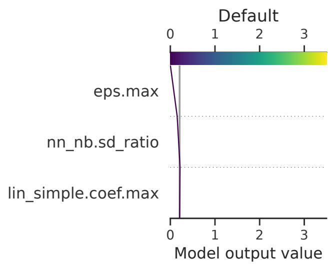
Figure 2: For each modCMA configuration, the $5d$ problem-instance footprint is displayed, with the y-axis marking the footprint regions and the x-axis representing the problem class ID ($f\_id$))

Most problem instances are well-solved by all modCMA configurations (Figure 1a), except for a few challenging ones for the poorest performer (as shown in the color map, where lower values denote better performance.). The clustering reveals nine distinct performance regions, driven by varying interactions among landscape features (Figure 1b). We can also conclude that most of the performance across the three algorithms falls within the fourth cluster, indicating that similar interactions of landscape features are driving this outcome. However, we can also observe that similar performance may appear in nearby clusters, driven by different landscape feature interactions. For example, instances in the first two clusters show similar performance but differ due to distinct landscape-feature interactions affecting algorithm predictions. This also holds for other clusters, such as the 3rd and 4th, where identical performance stems from different landscape dynamics.

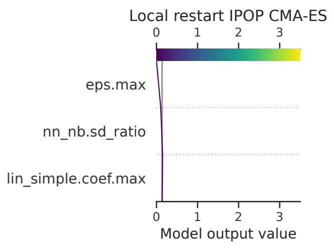
The clustering results, shown in Figure 2, highlight the footprints of the modCMA configurations. Figure 2 shows a "coverage matrix" with columns representing problem instances and rows indicating the algorithm and its corresponding cluster. The colors reflect the ground-truth performance on each algorithm instance, with clusters ordered by ascending average performance. Comparing the clusters assigned to the same problem instances across different algorithms reveals similarities or differences in their behavior. Assigning a problem instance to the first cluster under one algorithm and to the last cluster under another highlights a pronounced performance discrepancy between them. The first algorithm tackles the instance successfully, while the second finds it challenging. Furthermore, these similarities and differences can be quantified using similarity measures, such as cosine similarity applied to the problem instance meta-features. Consistent with expectations,



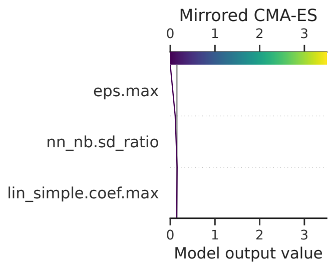
(a)



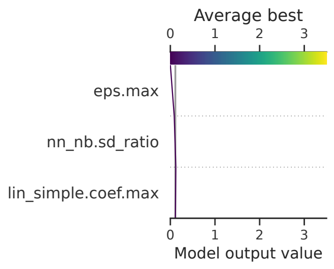
(b)



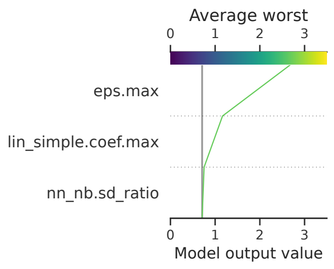
(c)



(d)

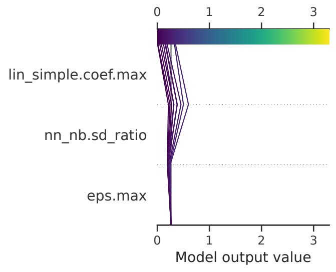


(e)

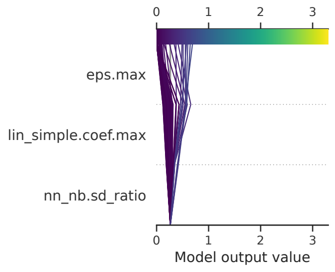


(f)

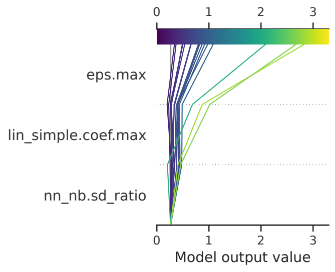
Figure 3: The subfigures depict the feature-importance profiles for second problem class across different modCMA-ES configurations on five-dimensional instances.



(a) Third cluster.



(b) Fourth cluster.



(c) Ninth cluster.

Figure 4: Feature importance patterns in the third, fourth, and ninth performance regions.

all configurations exhibit nearly identical footprint patterns, with the exception of the poorest-performing variant, which corresponds to the minimal differences observed in their ground-truth performance metrics in the raw data.

For a detailed examination, we centered our analysis on the problem class with ($f\_id = 2$). Figure 3 presents the dominant landscape features and their interrelations for this problem across every modCMA-ES variant, illustrating the interaction patterns that enable successful optimization in the five top-performing configurations and those that hinder the poorest-performing one. Each sub-figure displays a SHAP *decision plot*, showing each feature's contribution (SHAP value) to the model's prediction. The features on the y-axis are ordered by descending importance, with the topmost feature exerting the greatest influence. The x-axis displays the model's predicted value, and each curve shows how feature contributions accumulate from the baseline (mean ground-truth performance) to the final prediction. Curve colors encode predicted performance, ranging from dark blue (best) to yellow (worst). From this, it is evident that the five high-performing modCMA configurations share very similar contribution patterns, whereas these same feature interactions render the second problem especially difficult for the lowest-performing configuration, placing it in the poorest-performing region. This problem corresponds to the separable ellipsoidal function, and the set of modules used in the worst-performing configuration (see Table 1) fail to solve it. This problem fits the description of being unimodal and highly ill-conditioned. This would mean that we would expect huge values of lin_simple.coef.max (in the order of $10^6$), and ic.eps.max, while values close to one for nn_nb.sd_ratio. A recent fANOVA study [15] revealed that the "weights option" module plays a significant role in performance for that problem, both in isolation and through interactions with other modules. However, this is true only when the option is set to "default" or "lambda"; in our case, it is configured as "equal" weights. Additionally, another crucial module for this problem is the "mirrored" module [15]. The “Off” and “mirrored” options yield better results, while the “pairwise” option, used in our case, reduces performance. The combination of “pairwise” and “equal” further contributes to the worst performance. Additionally, as shown in Figure 3, the five best-performing configurations derive their most significant contribution from the eps.max feature, which enhances prediction accuracy, while the other two features contribute only marginally, staying close to the baseline. For the worst-performing configuration, both eps.max and lin_simple.coef.max contribute to predicting its poor performance, making the problem particularly challenging for the selected module options.

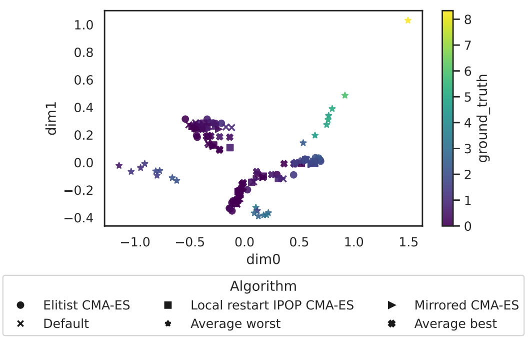
The worst-performing configuration also struggled with the ill-conditioned uni-modal problems (10, 11, and 12). The fANOVA study [15] confirms these patterns, as these problems are grouped with the second problem based on the contribution of the modCMA modules. Similar module option patterns, as observed in the second problem, are also evident here. While the previous study focused solely on the performance space, our approach also links these module interactions to landscape properties. For instance, eps.max identifies ill-conditioning by estimating the highest slope found, while lin_simple.coef.max reflects the highest coefficient in a linear model, providing similar insights. These features are shared by f10 (the rotated version of f2), f11 (the discuss function), f12 (the bent cigar function). They are unimodal with high condition numbers but with slight variations, e.g., the bent cigar is almost an ellipsoidal if the optimum is translated to the edges.

Due to space constraints, we cannot provide a detailed analysis of all problems. Instead, Figure 4 illustrates feature importance patterns in the third, fourth, and ninth performance regions. Each subplot contains problem instances paired with a configuration that belongs to that cluster (see Figure 1b). It is evident that the performance of configurations on problem instances in the third and fourth clusters is similar; however, these results are driven by different orders and interactions of key features. The ninth cluster represents the performance of configurations on problem instances that are challenging to solve, revealing the feature patterns responsible for this difficulty.

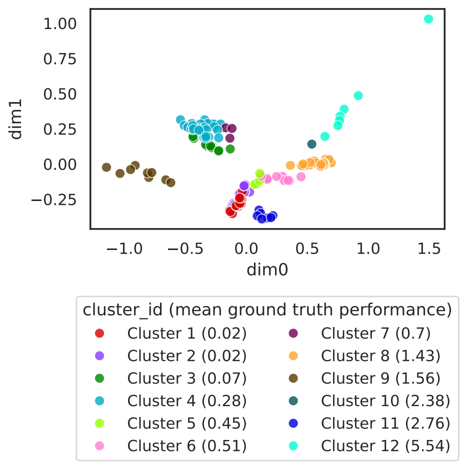
Figures 5 and 6 illustrate the performance of the modCMA configurations in 30-dimensional space. In particular, Figure 5 is divided into two panels: (a) the ground-truth performance (Figure 5a), and (b) the clustering results of the problem instance SHAP meta-features (Figure 5b). The clustering identifies 12 distinct performance regions, shaped by diverse interactions among landscape features (Figure 5b).

As in the $5d$ case, most problem instances are effectively solved by all modCMA configurations (see the footprint in Figure 6), except for those that pose challenges to the weakest performer (as indicated by the color map, where lower values represent better performance). Analysis on selected problems have been omitted due to the page limit.

Although the $5d$ and $30d$ footprints exhibit commonalities, these patterns are driven by each algorithm’s actual performance, good in lower dimensions does not necessarily carry over to higher ones.



(a)



(b)

Figure 5: The problem instances represented by their SHAP meta-features are projected into a two-dimensional vector space, with colors denoting: (a) the true performance, and (b) the cluster assignments of the SHAP meta-features for the thirty-dimensional ($30d$) test instances.

## 6. Discussion

By projecting various modular configurations into a joint space that captures the influence of landscape features on the performance of modCMA configurations, we can automatically identify distinct performance regions driven by different feature interactions. This analysis offers the advantage of avoiding subjective a priori thresholds to distinguish between good and poor performance. Instead, it uncovers all potential performance regions based on feature interactions, with performance gradually transitioning from best to worst.

By examining how the same problem instances are distributed across clusters under different configurations, we can determine which module combinations produce similar results (instances grouped in the same cluster) or different results (instances assigned to different clusters). Next, we can link these results to previous studies focusing solely on module contributions in the performance space, such as [15], [16], establishing a connection between the landscape and the module parameter space. Additionally, the landscape

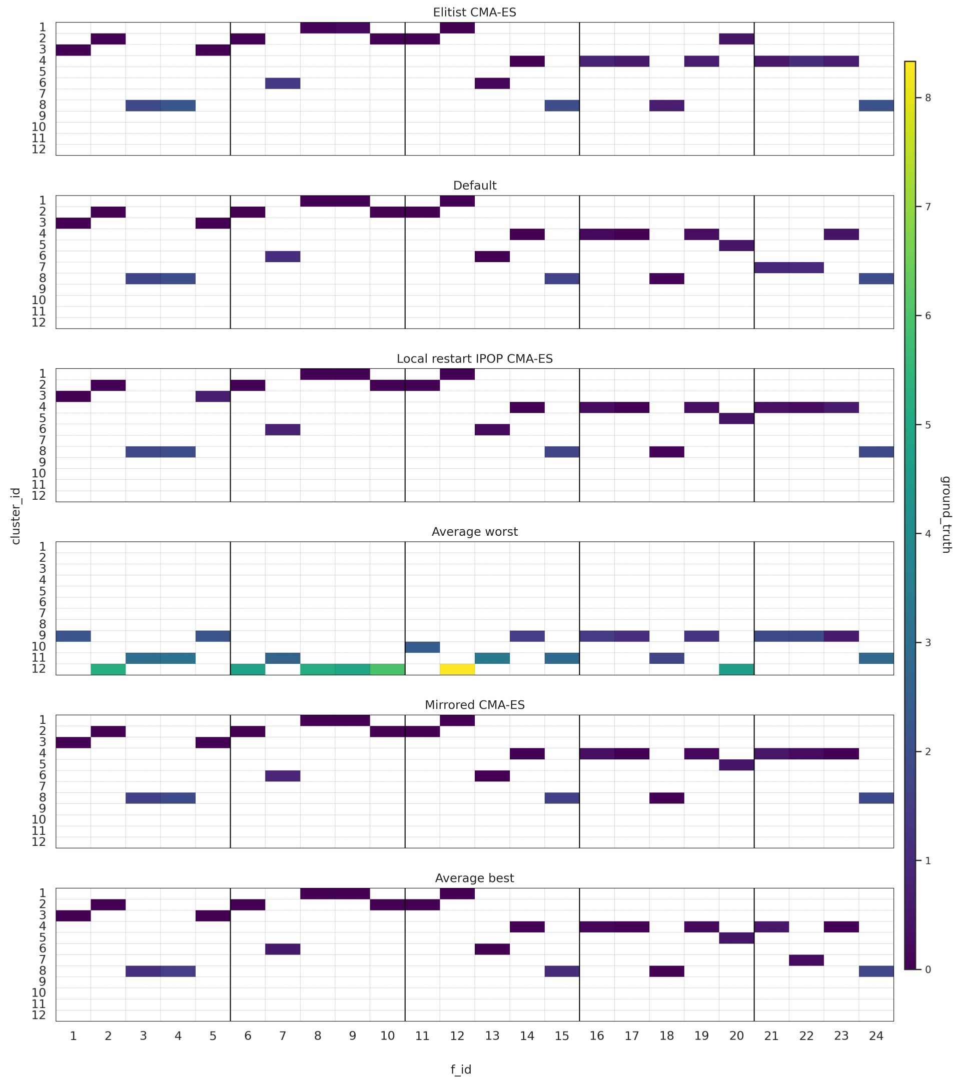


Figure 6: Displayed are the footprints of the $30d$ instances for each modCMA configuration, with footprint regions on the vertical axis and problem classes on the horizontal axis.

features relevant to each set of modules can be identified and analyzed for each problem individually.

Footprints can be computed at any stage of the optimization run, making it possible to compare different evaluation budgets and identify which landscape features are most influential at each point. They also extend naturally to fixed-target settings, where performance is defined by the number of evaluations needed to reach a specified goal instead of by proximity to the optimum.

That said, although multi-target regression is in theory capable of modeling many targets, its predictive power diminishes when the number of targets (i.e., algorithms) greatly exceeds the number of available features. This challenge remains open in the ML field, but with more configurations and larger problem sets (e.g., the MA-BBOB benchmark suite [34]), deep learning techniques could improve the MTR model to address a larger number of configurations.

Another limitation is the lack of diversity in the portfolio of six modCMA configurations studied, as the results indicate that five exhibit similar performance, with only one showing distinct behavior. In future work, we aim to perform a more in-depth analysis by employing automated methods [35] to select a complementary portfolio of diverse algorithm configurations to be analyzed.

Finally, we did not consider alternative clustering methods or clustering quality measures. Future work could explore different clustering strategies to enable a sensitivity analysis of the clustering step.

## 7. Conclusion

This work explores the utility of algorithm footprints in understanding the dynamic interplay between algorithm configurations and problem characteristics. Through the analysis of six modular CMA-ES (modCMA) variants on 24 BBOB benchmark problems, evaluated in both 5-dimensional and 30-dimensional settings, we uncovered key insights into performance variability and the problem-specific features driving it. The study revealed both shared behavioral patterns across configurations, stemming from common interactions with problem properties, and distinct behaviors on the same problem, influenced by differing features. As part of our future work, we aim to perform an in-depth analysis of each problem on an individual basis, using various modular frameworks such as modCMA, modDE, and PSO-X.

## Acknowledgment

We gratefully acknowledge the Slovenian Research and Innovation Agency for funding through program grant P2-0098, project grants J2-4460 and GC-0001, as well as the young researcher grant PR-12897 awarded to AN. This research also received support from the European Union under Grant Agreement No. 101187010 (HE ERA Chair AutoLearn-SI). M.A. Muñoz's contribution was funded by the Australian Research Council under grant IC200100009.

## References

[1] I. Bajaj, A. Arora, and M. F. Hasan, "Black-box optimization: Methods and applications," in *Black box optimization, machine learning, and no-free lunch theorems*. Springer, 2021, pp. 35–65.

[2] D. Molina, A. LaTorre, and F. Herrera, "An insight into bio-inspired and evolutionary algorithms for global optimization: review, analysis, and lessons learnt over a decade of competitions," *Cognitive Computation*, vol. 10, pp. 517–544, 2018.

[3] K. Varelas, A. Auger, D. Brockhoff, N. Hansen, O. A. ElHara, Y. Semet, R. Kassab, and F. Barbaresco, "A comparative study of large-scale variants of cma-es," in *Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part I 15*. Springer, 2018, pp. 3–15.

[4] M. Pant, H. Zaheer, L. Garcia-Hernandez, A. Abraham *et al.*, "Differential evolution: A review of more than two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103479, 2020.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. ieee, 1995, pp. 1942–1948.

[6] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[7] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss, "The sequential parameter optimization toolbox," in *Experimental methods for the analysis of optimization algorithms*. Springer, 2010, pp. 337–362.

[8] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, "Recent advances in selection hyper-heuristics," *European Journal of Operational Research*, vol. 285, no. 2, pp. 405–428, 2020.

[9] J. de Nobel, D. Vermetten, H. Wang, C. Doerr, and 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.

[10] S. Van Rijn, C. Doerr, and T. Bäck, "Towards an adaptive cma-es configurator," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2018, pp. 54–65.

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

[12] C. L. Camacho-Villalón, M. Dorigo, and T. Stützle, "Pso-x: A component-based framework for the automatic design of particle swarm optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 402–416, 2021.

[13] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *International conference on machine learning*. PMLR, 2014, pp. 754–762.

[14] M. Sundararajan and A. Najmi, "The many shapley values for model explanation," in *International conference on machine learning*. PMLR, 2020, pp. 9269–9278.

[15] A. Nikolikj, A. Kostovska, D. Vermetten, C. Doerr, and 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.

[16] N. van Stein, D. Vermetten, A. V. Kononova, and T. Bäck, "Explainable benchmarking for iterative optimization heuristics," *arXiv preprint arXiv:2401.17842*, 2024.

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

[18] A. Rasulo, K. Smith-Miles, M. Muñoz, J. Handl, and M. López-Ibáñez, "Extending instance space analysis to algorithm configuration spaces," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 147–150.

[19] A. Nikolikj, S. Džeroski, M. A. Muñoz, C. Doerr, P. Korošec, and T. Eftimov, "Algorithm instance footprint: Separating easily solvable and challenging problem instances," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 529–537.

[20] A. Nikolikj and T. Eftimov, "Comparing solvability patterns of algorithms across diverse problem landscapes," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 143–146.

[21] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, "Coco: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, vol. 36, no. 1, pp. 114–144, 2021.

[22] A. Nikolikj, M. A. Muñoz, and T. Eftimov, "Benchmarking footprints of continuous black-box optimization algorithms: Explainable insights into algorithm success and failure," *Swarm and Evolutionary Computation*, vol. 94, p. 101895, 2025.

[23] E. Korneva and H. Blockeel, "Towards better evaluation of multi-target regression models," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2020, pp. 353–362.

[24] A. Kostovska, D. Vermetten, P. Korošec, S. Džeroski, C. Doerr, and T. Eftimov, "Using machine learning methods to assess module performance contribution in modular optimization frameworks," *Evolutionary computation*, pp. 1–28, 2024.

[25] Q. Renau, C. Doerr, J. Dréo, and B. Doerr, "Exploratory landscape analysis is strongly sensitive to the sampling strategy," in *Proc. of Parallel Problem Solving from Nature (PPSN)*, ser. LNCS, vol. 12270. Springer, 2020, pp. 139–153. [Online]. Available: https://doi.org/10.1007/978-3-030-58115-2_10

[26] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 67, no. 2, pp. 301–320, 2005.

[27] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, 2016.

[28] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[30] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[32] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[33] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *ArXiv*, vol. abs/1109.2378, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:8490224

[34] D. Vermetten, F. Ye, T. Bäck, and C. Doerr, "Ma-bbob: A problem generator for black-box optimization using affine combinations and shifts," *ACM Transactions on Evolutionary Learning*, vol. 5, no. 1, pp. 1–19, 2025.

[35] A. Kostovska, G. Cenikj, D. Vermetten, A. Jankovic, A. Nikolikj, U. Skvorc, P. Korosec, C. Doerr, and T. Eftimov, "Ps-aas: Portfolio selection for automated algorithm selection in black-box optimization," in *International Conference on Automated Machine Learning*. PMLR, 2023, pp. 11–1.