Quantifying Module Interactions in the PSO-X Framework

Christian Leonardo Camacho Villalón[1,*] **Ana Nikolikj**[1,2,*] **Katharina Dost**[1] **Eva Tuba**[1,3,4]
Sašo Džeroski[1] **Tome Eftimov**[1]

[1]Jožef Stefan Institute, Slovenia
[2]Jožef Stefan International School, Slovenia
[3]Trinity University, TX, USA
[4]Singidunum University, Serbia
[*]Both authors serve as leading authors.

Abstract PSO-X incorporates dozens of algorithm components that have been proposed to solve single-objective continuous optimization problems using particle swarm optimization (PSO). While modular frameworks allow for flexible algorithm configuration and enable designers to automatically generate implementations tailored to specific optimization problems, understanding which modules matter most and how they interact remains an open question. In this study, we used performance data from 1,424 PSO algorithms instantiated from PSO-X and apply functional ANOVA to identify the impact that algorithm components, and their combinations, have on the algorithm performance tailored to different problem landscapes.

1 Introduction

In recent years, modular optimization frameworks, such as modCMA-ES (de Nobel et al., 2021), PSO-X (Camacho-Villalón et al., 2022) and modDE (Vermetten et al., 2023), have emerged as an effective approach for systematically designing, fine-tuning and assessing the performance of randomized optimization algorithms. These frameworks allow researchers to generate optimization algorithms from discrete modules, which are often interchangeable and responsible for specific algorithmic behaviors (Camacho-Villalón et al., 2023; Tzanetos and Kudela, 2024). There are three main advantages to using modular optimization frameworks: (i) they enable the use of an automatic design approach, in which automatic algorithm configuration tools (e.g., irace (López-Ibáñez et al., 2016) and SMAC (Hutter et al., 2011)) explore the framework's design space in order to find high-performing designs; (ii) they facilitate fair comparisons across the algorithmic variants included in the framework on different benchmarks (Doerr et al., 2018); and (iii) they promote reproducibility by providing a standardized implementation of algorithmic variants (Bartz-Beielstein et al., 2020).

Despite the clear advantages that modular optimization frameworks have at a practical level, they often include dozens or even hundreds of implementation options, which make it challenging to fully understand each module's contribution to the overall performance of the resulting algorithms. In this study, we conducted a comprehensive investigation of module importance and their interactions within the PSO-X framework across varied problem landscapes.

2 Related Work

Previous research on the performance of modular optimization frameworks has primarily studied module importance via a frequency-based analysis of top-performing configurations. In (de Nobel et al., 2021; Vermetten et al., 2023), the authors used the automatic algorithm configuration tool irace to identify commonly selected modules in successful configurations of modCMA-ES and modDE. Few papers have focused on studying modules contribution in a more systematic manner. For example, in (Kostovska et al., 2024, 2022), the authors used problem landscape features to

train an ML regression model to predict the optimal module selection, whereas in (Van Stein et al., 2024), the authors applied XAI techniques to analyze module importance. These works, however, only consider individual module interaction. To our knowledge, the sole attempt to measure how much variance each module—individually or in combination—adds to overall performance within modular frameworks such as modDE and modCMA-ES is the work by (Nikolikj et al., 2024). Yet no comparable variance-decomposition analysis has been performed for the PSO-X framework within different problem landscapes. This paper therefore fills an important gap in the literature.

3 Methodology for Quantifying Module Effects

To uncover problem groups that share similar patterns of module contributions and interactions in PSO-X, we apply functional ANOVA (f-ANOVA) (Hutter et al., 2014; Van Rijn and Hutter, 2018). Originating in ML, f-ANOVA analyzes automatic configuration tools by partitioning performance variance into main, pairwise, and higher-order effects (module interactions), yielding precise, statistically grounded insights into which modules and combinations of modules matter most in modular optimization frameworks. For each benchmark function, we assemble a dataset that lists every PSO-X variant tested on it, encoding the variant's module settings as features and its achieved performance as the target. The datasets are then given as input to f-ANOVA to quantify how much individual modules and their combinations influence performance in the specific problem class. Our approach can be summarized as follows:

1) Quantifying module effects with f-ANOVA: The datasets serve as input to f-ANOVA, which decomposes the variance in performance and attributes it systematically to individual modules and their interactions. Given n modules that define the variants, the analysis quantifies the contribution of each of the n individual modules, as well as all $\binom{n}{2}$ pairwise and $\binom{n}{3}$ triple module interactions. This process is repeated separately for each problem-class-specific dataset.

2) Grouping problem classes based on module effects: To analyze how module effects differ across problem classes, the quantified effects are first concatenated into a single vector representation for each class (i.e., can be assumed as an embedding). Each vector has a dimensionality of $n + \binom{n}{2} + \binom{n}{3}$, capturing the individual, pairwise and triple interaction effects. Once these vectors are constructed, a clustering algorithm is applied to identify groups of problem classes that share similar module effect patterns and distinguish those that differ. This analysis is conducted in the original high-dimensional space to avoid any potential loss of information that may result from dimensionality reduction techniques. Next, we examine how the clusters identified in the previous step correspond to established groupings of problem classes based on high-level characteristics, such as modality or conditioning.

4 Experimental Design

The PSO-X framework (Camacho-Villalón et al., 2022) is a modular implementation of the particle swarm optimization (PSO) algorithm (Kennedy and Eberhart, 1995; Eberhart and Kennedy, 1995). The framework incorporates a wide range of modules inspired by various state-of-the-art PSO variants, where key design choices have been translated into distinct modules. This modular setup enables the construction of thousands of unique PSO-X variants. To systematically generate the PSO variants considered in our study, we took the Cartesian product of eight modules and 26 implementation options (see Table 1 in the Supplemental Section A), resulting in a total of 1424 variants. Each algorithm variant was executed independently 10 times on each of the 25 problem classes from the CEC'05 benchmark suite. The evaluation budget was set to $2500d$ function evaluations, where d is the number of dimension of the optimization problem. Performance was assessed based on the distance between the best-found solution and the known global optimum (*distance*). To ensure numerical stability and meaningful comparisons, distances smaller than 10^{-9} were capped at this threshold. For each problem instance, the median distance across the 10 runs

was taken as the final performance measure. These values were then log-transformed using base 10. As a result, the *distance* metric has a lower bound of -9, with lower values indicating better optimization performance.

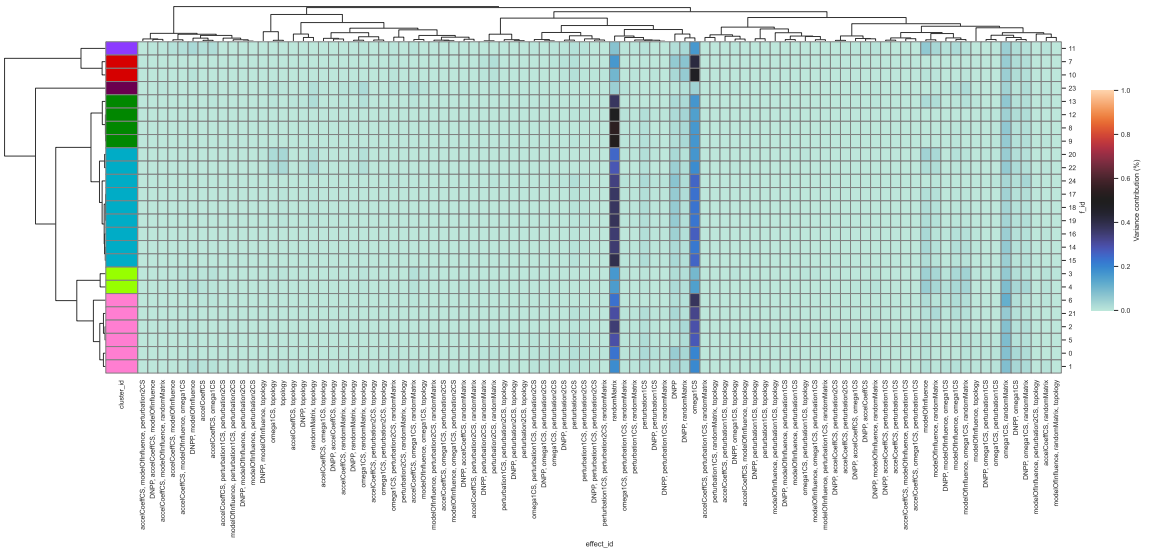


Figure 1: Clustermap. The color-coding in the first column of the clustermap indicates the cluster. The rows are the problem classes, while the moduls interactions are represented on x-axis.

The experiments are conducted on the CEC'05 benchmark suite (Suganthan et al., 2005), which comprises 25 problem classes, including unimodal, multimodal and hybrid composition functions. We consider only problems defined in a ten-dimensional space (i.e., with $d = 10$). The performance distribution of the 1,424 PSO-X variants across the 25, ten-dimenisonal problem classes from the CEC'05 suite is presented in Supplemental (see Section B).

This study aims to quantify the importance of individual modules for each of the 25 problem classes. To do so, the data is first organized into 25 separate datasets, one for each problem class. Each dataset contains 1,424 algorithm variants (data instances), where each instance is described by eight module settings (features), and the target variable corresponds to the performance of the variants on the respective problem class. Running f-ANOVA on each dataset individually results in 25 vector representations, each capturing the module effects through $8 + \binom{8}{2} + \binom{8}{3}$ terms. To identify similarities among problem classes based on these effects, we apply Hierarchical Clustering (HC) (Müllner, 2011). Given the relatively small dataset (25 instances) and our focus on interpretability, HC was chosen for its ability to provide a clear and transparent clustering process via a dendrogram, which visually illustrates how clusters are formed and merged at each stage. To determine the most suitable number of clusters, we conduct a grid search over the clustering algorithm's hyperparameters and evaluate the results using the Silhouette coefficient (Rousseeuw, 1987). This metric, ranging from -1 to 1, reflects the quality of clustering—where higher values indicate more compact and well-separated clusters. The hyperparameter settings explored during this search are detailed in Table 2 in the Supplemental A.

5 Results and discussion

We selected seven clusters (see Supplemental C), the *cosine* distance metric, and the *complete* linkage method, for the clustering experiment. While clusters are still distinguishable (as supported by the moderate Silhouette score of 0.55), the low inter-cluster distance of 0.065 implies that the module importance patterns are quite similar overall, and the boundaries between clusters are

subtle. Figure 1 presents a clustermap of the vector representation. Rows correspond to problem classes (f_id), and columns to module effects ($effect\_id$). The color intensity indicates the variance contribution of each module effect, ranging from 0 (no importance) to 1 (high importance). The first column shows cluster assignments, with each color representing a different cluster. The results reveal that most problem classes fall into three dominant clusters—green, cyan, and pink—while a few remain isolated in smaller, distinct clusters. The clustermap reveals that the individual effects of the *randomMatrix* and *omega*1*CS* modules, and to a lesser extent *DNPP*, are the most influential across problem classes. Among the interaction effects, combinations involving these modules—such as *omega*1*CS* + *randomMatrix*, *DNPP* + *omega*1*CS*, *DNPP* + *randomMatrix*, and the triplet *DNPP* + *omega*1*CS* + *randomMatrix*—stand out as highly impactful. This suggests that the behavior of the PSO-X variants is primarily driven by the interplay of these three modules.

These findings are consistent with theoretical and experimental studies of PSO variants (Bonyadi and Michalewicz, 2017). In PSO-X, *omega*1*CS* specifies the strategy for controlling particle inertia and plays a direct role in the local convergence of the algorithm, whereas *randomMatrix* ensures the implementation is rotation-invariant. As the vast majority of the CEC'05 functions are multimodal and/or rotated, *omega*1*CS* and *randomMatrix* become critical to the algorithm design. Conversely, the *DNPP* module determines how particles combine the position vectors of other particles in the swarm, as well as their personal best and velocity vectors. The influence of the *DNPP* module largely depends on the *omega*1*CS* and *randomMatrix* modules being enabled.

Figure 4, in Supplemental D, displays the marginal performance of the options for the most important module effects, namely the *randomMatrix* and *omega*1*CS* modules, in problem classes with $f\_id \in \{0, 8, 11\}$. The problem class was selected randomly, one per the three larger clusters. In Figure 4, 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). The heatmaps (Figure 4) presented, display the marginal performance for a combination of module options, the x and y-axis show the different options, while the color-coding indicates the marginal performance, where lower values indicate better performance (i.e. lower distance to the optimum). The 0, 1 and 4 options for *randomMatrix* consistently yield the best marginal performance across all problem classes. For the *omega*1*CS* the 0.75, 12 and 14 options tend to result in better marginal performance. When analyzing the pairwise interactions between *randomMatrix* and *omega*1*CS*, these high-performing individual options also combine well.

6 Conclusions

Although we can present only a subset of our findings here—these are preliminary results, trimmed for space, with higher-dimensional cases omitted—our f-ANOVA study of 1,424 PSO-X variants already demonstrates which individual modules drive performance and how their interactions shape output across diverse landscapes. By clustering problems with similar variance-decomposition profiles, we expose recurring design patterns and pinpoint context-dependent synergies among modules. In the future, these insights will provide principled guidance for composing PSO components and set the stage for more adaptive, landscape-aware optimization strategies.

References

Acknowledgements.

Bartz-Beielstein, T., Doerr, C., Berg, D. v. d., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., et al. (2020). Benchmarking in optimization: Best practice and open issues. *arXiv preprint arXiv:2007.03488*.

Bonyadi, M. R. and Michalewicz, Z. (2017). Particle swarm optimization for single objective continuous space problems: a review.

Camacho-Villalón, C. L., Dorigo, M., and Stützle, T. (2022). PSO-*X*: A component-based framework for the automatic design of particle swarm optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 26(3):402–416.

Camacho-Villalón, C. L., Stützle, T., and Dorigo, M. (2023). Designing new metaheuristics: Manual versus automatic approaches. *Intelligent Computing*, 2(0048):1–15.

de Nobel, J., Vermetten, D., Wang, H., Doerr, C., and Bäck, T. (2021). Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules. In Chicano, F., editor, *GECCO'21 Companion*, pages 1375–1384, New York, NY. ACM Press.

Doerr, C., Wang, H., Ye, F., Van Rijn, S., and Bäck, T. (2018). IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv preprint arXiv:1810.05281*.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Piscataway, NJ. IEEE Press.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In Coello Coello, C. A., editor, *Learning and Intelligent Optimization, 5th International Conference, LION 5*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer, Heidelberg, Germany, Heidelberg, Germany.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32, pages 754–762.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948, Piscataway, NJ. IEEE.

Kostovska, A., Vermetten, D., Džeroski, S., Doerr, C., Korosec, P., and Eftimov, T. (2022). The importance of landscape features for performance prediction of modular cma-es variants. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 648–656.

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

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Nikolikj, A., Kostovska, A., Vermetten, D., Doerr, C., and Eftimov, T. (2024). Quantifying individual and joint module impact in modular optimization frameworks. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., and Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore.

Tzanetos, A. and Kudela, J. (2024). Working on the structural components of evolutionary approaches. In *16th International Joint Conference on Computational Intelligence ECTA, Porto, Portugal, 20-22 November 2024*, pages 375–382. Science and Technology Publications, Lda.

Van Rijn, J. N. and Hutter, F. (2018). Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2367–2376.

Van Stein, N., Vermetten, D., V. Kononova, A., and Bäck, T. (2024). Explainable benchmarking for iterative optimization heuristics. *ACM Transactions on Evolutionary Learning*.

Vermetten, D., Caraffini, F., Kononova, A. V., and Bäck, T. (2023). Modular differential evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 864–872, New York, NY. ACM Press.

A Supplemental Section

Table 1: The eight PSO-X modules and 26 implementation options considered in this work.

module_id	option_id
DNPP	DNPP-rectangular, DNPP-spherical, DNPP-standard, DNPP-Gaussian, DNPP-discrete, DNPP-Cauchy-Gaussian (operatorCG_parm_r 0.5)
AC	AC-constant (phi1 1.4, phi2 1.4), AC-random (initialPhi1 2.4, finalPhi1 0.5, initialPhi2 0.5, finalPhi2 2.4)
Topology	Top-ring, Top-fully-connected
MoI	MoI-best-of-neighborhood, MoI-fully informed
randomMatrix	none, Mtx-random diagonal, Mtx-Euclidean rotation (angleCS 2, angle_par_alpha 30, angle_par_beta 0.01), Mtx-Increasing group-based
omega1CS	IW-constant (omega1 0.75), IW-constant (omega1 0.0), IW-adaptive based on velocity (iw_par_lambda 0.5, initialIW 0.15, finalIW 0.95), IW-rank-based (initialIW 0.15, finalIW 0.95), IW-success-based (initialIW 0.15, finalIW 0.95)
perturbation1CS	none, Pertinfo-Gaussian (magnitude1CS 4, magnitude1 0.5, mag1_par_success 40, mag1_par_failure 20)
perturbation2CS	none, Pertrand-rectangular (magnitude1CS 4, magnitude1 0.5, mag1_par_success 40, mag1_par_failure 20)

We show in parentheses the default values for the parameters associated with specific implementation options.

Table 2: Hyperparameter search space for the HC algorithm.

Hyperparameter	Range
n_clusters	{2, ..., 25}
metric	{cosine, euclidean}
linkage	{single, complete, average, ward}

B Algorithm performance distribution

This section analyzes how the PSO-X variants perform in each of the problem classes. To measure performance, we use the *distance* metric, which captures the gap between the best-found solution (within a certain budget of function evaluations) and the global optimum. The distance values are aggregated using the median across multiple runs on the same problem class, followed by a logarithmic transformation as outlined in the experimental setup.

Figure 2 shows the performance distribution of 1,424 PSO-X variants across the 25, ten-dimensional problem classes from the CEC'05 suite. The spread of the boxplots indicates variability in performance, highlighting the potential for algorithm configuration. The higher variability in problem classes with $f\_id \in \{0-5, 8, 9, 11, 12\}$ suggests that the choice of variant significantly impacts performance. In contrast, low variability in classes with $f\_id \in \{7, 10, 13-24\}$ implies that majority of the variants perform very similarly.

C Clustering

Figure 3 displays the clustering results in two sub-figures. The top-left panel of Figure 3a shows the tuning outcomes for different clustering settings, varying the number of clusters, distance metrics, and linkage methods. The x-axis denotes the number of clusters, while the y-axis shows the Silhouette coefficient (ranging from -1 to 1), where higher values indicate more coherent clusters. Each line represents a different combination of hyperparameters. Figure 3b presents a dendrogram—a tree-like diagram representing the hierarchical clustering process. It visualizes

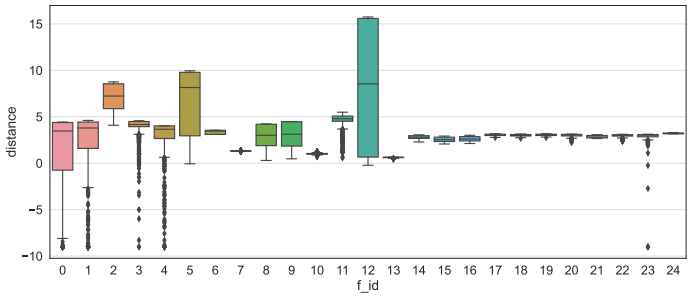
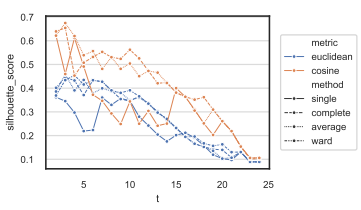
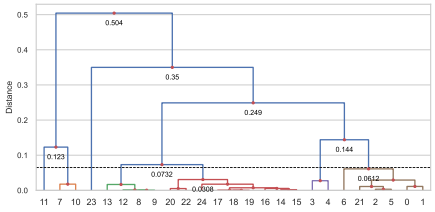


Figure 2: Performance distribution of 1,424 PSO-X variants on 25 problem classes comprising the CEC2005 suite (10d), evaluated using the *distance* metric—the absolute difference between the best-found solution (within 25,000 evaluations) and the global optimum. The performance values are presented on a logarithmic scale.

how problem classes are iteratively grouped based on similarity, with branch heights indicating inter-cluster dissimilarity. Color coding highlights the resulting clusters.



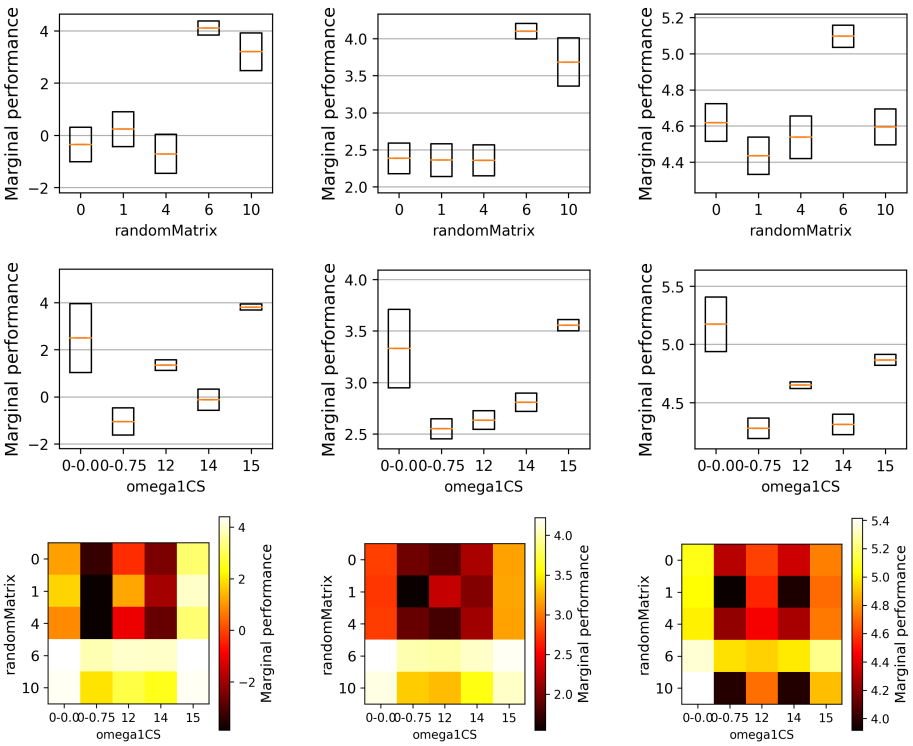
(a) Performance results from the tuning of the HC algorithm. Silhouette coefficient is used as clustering quality measure.



(b) Dendrogram illustrating the clustering process of HC.

Figure 3: Clustering results of the 25, ten-dimensional problem classes from the CEC2005 suite based on the vector representations generated with f-ANOVA.

D Marginal performance



(a) $f\_id = 0$ (b) $f\_id = 8$ (c) $f\_id = 11$

Figure 4: Marginal performance of the options for the *randomMatrix* and *omega1CS* modules, in problem classes with $f\_id \in \{0, 8, 11\}$. Sub-figures (a) - (c) show the individual effects of the modules (first two rows, respectively, while the third row illustrates their pairwise interaction.