Quantifying Individual and Joint Module Impact in Modular Optimization Frameworks

Ana Nikolikj, Ana Kostovska, Diederick Vermetten, Carola Doerr, Tome Eftimov

Quantifying Individual and Joint Module Impact in Modular Optimization Frameworks

Ana Nikolikj, Ana Kostovska, Diederick Vermetten, Carola Doerr and Tome Eftimov

***Abstract*—This study explores the influence of modules on the performance of modular optimization frameworks for continuous single-objective black-box optimization. There is an extensive variety of modules to choose from when designing algorithm variants, however, there is a rather limited understanding of how each module individually influences the algorithm performance and how the modules interact with each other when combined. We use the functional ANOVA (f-ANOVA) framework to quantify the influence of individual modules and module combinations for two algorithms, the modular Covariance Matrix Adaptation (modCMA) and the modular Differential Evolution (modDE). We analyze the performance data from 324 modCMA and 576 modDE variants on the BBOB benchmark collection, for two problem dimensions, and three computational budgets. Noteworthy findings include the identification of important modules that strongly influence the performance of modCMA, such as the *weights option* and *mirrored* modules for low dimensional problems, and the *base sampler* for high dimensional problems. The large individual influence of the *lpsr* module makes it very important for the performance of modDE, regardless of the problem dimensionality and the computational budget. When comparing modCMA and modDE, modDE undergoes a shift from individual modules being more influential, to module combinations being more influential, while modCMA follows the opposite pattern, with an increase in problem dimensionality and computational budget.**

***Index Terms*—meta-learning, single-objective optimization, module importance**

I. Introduction

A core application of evolutionary algorithms is the optimization of continuous single-objective black-box problems. A plethora of algorithms already exist [1], [2], and as the field matures researchers are increasingly focusing on refining existing algorithms rather than developing new algorithmic paradigms [3]–[6]. For example, in the data from the BBOB workshops [7] we can find more than 50 variants of the CMA-ES algorithm [8].

Ana Nikolikj (Email: ana.nikolikj@ijs.si) and Tome Eftimov (Email: tome.eftimov@ijs.si) are with Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia.

Ana Kostovska (Email:ana.kostovska@ijs.si) is with the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia.

Ana Nikolikj. and Ana Kostovska are also with the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

Diederick Vermetten (Email: d.l.vermetten@liacs.leidenuniv.nl) is with the Leiden University, LIAC, Leiden, Netherlands.

Carola Doerr (Email: carola.doerr@lip6.fr) is with the Sorbonne Université, CNRS, LIP6, 1000 Paris, France.

The authors acknowledge the support of the Slovenian Research Agency through program grants P2-0098 and P2-0103, young researcher grants No. PR-12897 to AN and PR-09773 to AK, project No. J2-4460, and a bilateral project between Slovenia and France grant No. BI-FR/23-24-PROTEUS-001 (PR-12040).

Rather than studying algorithm variants in isolation, it was proposed in [4], [5] to consider a standardized *modular optimization framework* where various algorithm variants can be systematically assessed, ensuring uniformity in implementation details across all variants. The main idea behind modular optimization frameworks is to break down an algorithm into smaller components called *modules*, which can then be configured by taking on different options that influence the algorithm's behavior. The modules can be seamlessly integrated to form new algorithm variants. Additionally, each module operates independently, so it can be removed without affecting the functionality of the rest. The above-mentioned studies emphasize the advantages of modular algorithms, both in terms of fair comparisons from an implementation perspective and also the possibility of examining the interactions between different modules.

The challenge that remains is how to quantify the influence of individual modules and the influence of module interactions on the algorithm performance. Often, algorithm designers manually assess module influence by investigating the local neighborhood of a given algorithm variant, for example, a common approach is to alter one module at a time and to then observe the changes in performance. Probably even more common is to simply compare the performance of an algorithm variant to a default implementation of the same algorithm, or compare it to a limited set of other variants. The only information obtained with this analysis is how the different module options perform in the context of a few other variants, analyzing only the individual influence of a module and ignoring the interaction influence that results from the combination with the other modules.

Our contribution: In this study, we employ the f-ANOVA framework proposed in [9] to quantify individual and interaction module effects in modular optimization frameworks. Our study involves the performance data of the modCMA [4] and modDE [5] frameworks, evaluated on the 24 problems from the BBOB benchmark suite, in dimensions 5 and 30, for three different computational budgets. By analyzing 324 variants of modCMA and 576 variants of modDE, we assess individual and interaction module effects through two scenarios: one evaluating the module's effects on the level of an entire benchmark suite and another examining their effects on a problem-specific level. The analysis reveals important modules and combinations of modules, such as *weights option* and *mirrorred* in modCMA for dimension 5 and *base sampler* for dimension 30. Additionally, it highlights the importance of

individual modules such as *lpsr*, over module combinations in modDE, particularly for lower budgets. Overall, the findings provide valuable insights for algorithm designers and suggest potential applications for problems where module combinations contribute to algorithm performance.

Outline: Section II reviews relevant research on module importance in modCMA and modDE, together with studies from related domains that utilize f-ANOVA to estimate the importance of hyperparameters. Section III outlines the functional ANOVA methodology. The experimental design is explained in Section IV, followed by a discussion of key results in Section V. Section VI concludes the paper.

Data and code availability: The data and the code involved in this study are available at [10].

II. Related work

Initial work in this direction [11] involves conducting a comprehensive assessment of the importance of modules in CMA-ES. This was achieved by utilizing data from a complete enumeration of all module options to analyze the performance contribution of each module. However, this method proves unfeasible when dealing with an increasing set of modules.

Both [4] and [5] utilize an algorithm tuning tool called *irace* [12] to explore a wide space of possible algorithm variants, with the goal to identify a set of *elite configurations* that perform best over a set of optimization problems. Then, the module importance is represented as the module's frequency in the set of elite configurations. This process is repeated by iteratively adding modules to the space *irace* explores. The outcomes emphasize that by integrating modules, the dynamics between the modules underwent changes, which became apparent through visualization of the results from their frequencies in the elite configurations. Nevertheless, a comprehensive quantitative analysis isolating all effects of the modules (i.e., including individual, pairwise, triple effects, etc.) is missing.

Assessing the importance of various modules for the performance of modular optimization algorithms is analogous to evaluating the importance of hyperparameters in Machine Learning (ML) models. A seminal work in this domain is [9]. Their approach is built on top of the classic technique of functional analysis of variance, previously introduced for high-dimensional functions of dependent variables [13]. To quantify the influence of hyperparameters and their interactions in ML algorithms, [9] expands the concept of the functional-ANOVA (f-ANOVA) framework as an efficient linear-time algorithm for calculating the interaction effects of hyperparameters. The efficacy of this approach was then used to investigate highly parametric ML frameworks and combinatorial solvers designed to optimize NP-hard problems.

III. Assessing module importance with f-ANOVA

f-ANOVA decomposes the observed variance V of a response variable $f : \Theta_1 \times \cdots \times \Theta_n \to \mathbb{R}$ (in our case: algorithm performance) into additive components V_U, attributing each component to a corresponding subset of the inputs $U \subseteq \{1, \ldots, n\}$ (in our case: algorithm modules), and guarantees that the components add up the total variance of f in the end:

$$V = \sum_{U \subseteq \{1,\ldots,n\}} V_U \tag{1}$$

Fully decomposed, we can represent the variance in algorithm performance as:

$$\begin{aligned} V = V_1 + \cdots + V_n \\ + V_{1,2} + \cdots + V_{1,n} + \cdots + V_{n-1,n} \\ + \cdots \\ + V_{1,\ldots,n} \end{aligned} \tag{2}$$

The components V_U for $|U| = 1$ are called *individual effects* and represent the variance in the algorithm performance caused by varying a single module's options. The components for $|U| > 1$ capture the *interaction effects* between all modules in U. They represent the variance in the algorithm performance caused by varying the options for all the modules in U.

To obtain this decomposition, we must be able to efficiently compute the effects over arbitrary sets of modules. In [9] the authors provide an efficient and exact method for calculating the individual, pairwise, and triple interaction effect of algorithm hyperparameters by performing variance decomposition using a random forest model [14]. In our case, the hyperparameters are analogous to the modules.

IV. Experimental design

In this section, we provide details on the experimental setup. We first describe our data including the algorithm portfolio, the problem portfolio, and the algorithm performance data. Next, the datasets used by the f-ANOVA approach for quantifying module efefcts are presented in more detail.

A. Experimental data

We use publicly available data from the study presented in [15]. This repository contains performance data of 324 modCMA variants and 576 modDE variants, generated with the modular frameworks from [4] and [5], respectively. See Tables I and II for a summary of the six modCMA modules, and Tables IV and III for the seven modDE modules considered in our study. Using the IOHexperimenter platform [16], each variant is run 10 independent times on the first five instances of each of the 24 single-objective, noiseless black-box optimization problems from the BBOB benchmark suite [17] on the COCO platform [18], in dimension $d \in \{5, 30\}$. For each run on each problem instance, we calculate the *target precision* defined as the difference between the best-obtained solution and the global optimum. We consider the median target precision of the ten runs obtained after a budget of $100d$, $500d$, and $1500d$ function evaluations, respectively. Next, we transform the median target precision for each variant and each problem instance into logarithmic space, which is further used in our experiment referred to as the *solution precision*.

TABLE I: Short description of the role of modules in modCMA.

Module	Description
1. base_sampler	The mechanism for sampling new candidate solutions around the current estimate of the optimum.
2. elitism	The mechanism ensures that the best individual/s from the current population is preserved into the next generations.
3. local_restart	If the optimization process stagnates, a "restart" is performed from a different point in the search space.
4. mirrored_sampling	For every new sampled solution its mirrored image is added to the population.
5. weights_option	Recombination weights control how solutions from the current population are merged to create new individuals.
6. step_size_adaptation	Strategy for adapting the algorithm's parameters which control the search process.

TABLE II: Available options for the six modCMA modules evaluated in our experiments (324 combinations in total).

Module	Options
1. base_sampler	Gaussian, Sobol, Halton
2. elitist	off, on
3. local_restart	off, IPOP, BIPOP
4. mirrored_sampling	off, mirrored, mirrored pairwise
5. weights_option	default, equal, (1/2)^λ
6. step_size_adaptation	csa, psr

TABLE III: Available options for the seven modDE modules evaluated in our experiments (576 combinations in total).

Module	Options
1. adaptation_method	off, shade, jDE
2. crossover	bin, exp
3. lpsr	off, on
4. mutation_base	rand, best, target
5. mutation_n_comps	1, 2
6. mutation_reference	off, pbest, best, rand
7. use_archive	off, on

B. f-ANOVA datasets

To quantify the individual and combined importance of modules in modular frameworks, we perform two distinct use-cases: one across the entire benchmark suite and another at the problem-specific level.

Benchmark suite-level: We compute the average solution precision for each variant across all 120 problem instances. This average solution precision serves as an estimation of the variant's performance across the entire BBOB benchmark suite. Subsequently, for modCMA, each variant is represented using six features (refer to the modules in Table II) and associated with the achieved mean solution precision. Likewise, for modDE, each variant is represented through seven features (refer to the modules in Table III) and linked to the achieved mean solution precision across the complete benchmark suite. Following these transformations, we generate 12 datasets, two modular frameworks $\times$ two problem dimensions ($d = 5, 30$) $\times$ three different budgets of function evaluations ($100d, 500d, 1500d$), encompassing 324 variants for modCMA and 576 variants for modDE.

Problem-level: For this experiment we calculate the median solution precision of each variant, considering all five problem instances within each problem class. Following this, we generate distinct datasets for each problem by isolating the performance outcomes specific to that particular problem in two distinct problem dimensions ($d = 5, 30$). Consequently, we obtain 24$\times$2 datasets for each modular framework individually. Next, we repeat this for three different budgets (288 datasets in total). Similar to the benchmark suite-level approach, in the modCMA datasets, variants are characterized by six features associated with the module options, correlating with the median solution precision achieved on that specific problem. Conversely, in the case of modDE, seven features are utilized.

C. f-ANOVA

For variance decomposition analysis with a random forest model, we employed the implementation from [9]. This method involves utilizing a random forest model as a regression tree predictor to forecast the performance of a variant using different module options. The complete dataset serves as the training data, upon which variance decomposition is directly applied using the trees within these forests. For a comprehensive understanding of why the training scenarios (such as cross-fold validation, or train-validation-test split) of the RF model are excluded, please refer to the original paper where detailed explanations are provided [9].

V. Results and discussion

Here, the findings are categorized into two use cases: benchmark suite-level and problem-level. Within each use case, we display the outcomes of individual module effects, pairwise interaction effects, and triple interaction effects, for modCMA and modDE respectively, spanning two distinct problem dimensions and three different budgets.

A. Benchmark suite-level

Table V displays the cumulative fraction of variance in the algorithm performance explained by the individual, pairwise, and triple module effects separately and provides information about how different types of module effects contribute to the total explained variance shown by the column "total" in the table. The cumulative individual effect results from summing the individual effects of six modules for modCMA and seven modules for modDE, the cumulative pairwise effect is summed over $\binom{6}{2} = 15$ module pairs of modCMA and $\binom{7}{2} = 21$ of modDE, and finally, the cumulative triplet effect is calculated over $\binom{6}{3} = 20$ module triples for modCMA and $\binom{7}{3} = 35$ for modDE. The table contains the results for both algorithms modCMA and modDE, when considering two problem dimensions $d = 5$ and $d = 30$ and three different budgets $100d$, $500d$, and $1500d$. The columns in the table are named correspondingly to the different effect types and the values denote % of the total variance explained.

Focusing on how the cumulative individual, pairwise, and triple effects vary across the different scenarios for modCMA, in $d = 5$, the individual and pairwise effects are relatively

TABLE IV: Short description of the role of modules in modDE.

Module	Description
1. adaptation_method	The strategy for adapting the algorithms' parameters which control the search process.
2. crossover	Defines how the offspring solutions are generated from the parent solutions.
3. lpsr	Strategy to gradually decrease the size of the population as the algorithm progresses.
4. mutation_base	This determines the base solution used in the mutation process, affecting the diversity of the population.
5. mutation_n_comps	This parameter specifies the number of components (difference vectors) used in the mutation step.
6. mutation_reference	Determines which individuals are used as references in the mutation process.
7. use_archive	Controls whether past solutions are used in the evolution process.

balanced, while the triplet effect is smaller, for all budgets. In higher dimensionality ($d = 30$) the individual effects tend to amplify.

Next, we will look at how the cumulative individual, pairwise, and triple effects vary for modDE. As the budget increases, the cumulative individual effect tends to decrease gradually, while the pairwise and triple effects increase correspondingly. This trend is consistent across both dimensions. Furthermore, comparing the same type of effects at different dimensions (5 vs. 30) for the same budget, there is a noticeable decrease in the individual effects and an increase in the pairwise and triple effects as the dimensionality increases. For the largest budget, the individual and pairwise effects even become balanced. Overall, for modDE, these observations indicate that the interactions between the modules (pairwise and triple effects) play a more significant role in the algorithm's performance as both the problem dimensionality and the budget for the algorithm increase.

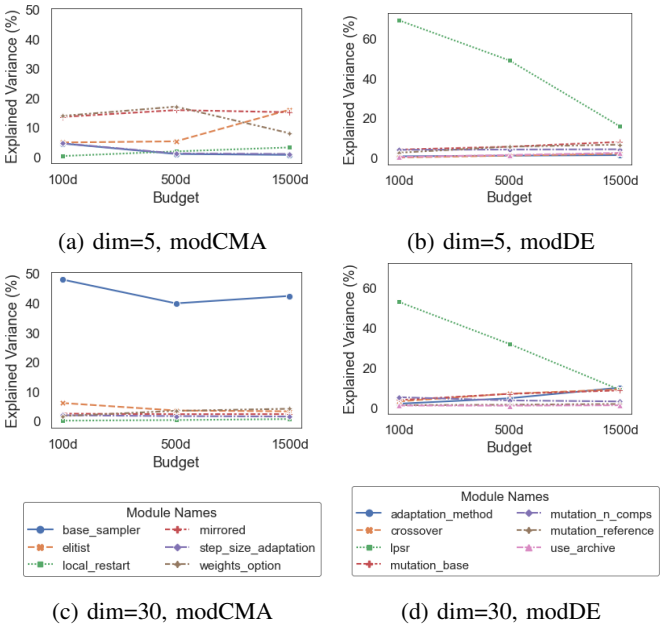
Comparing modCMA and modDE, while both algorithms exhibit different shifts in the importance of individual, pairwise, and triple effects with changes in both dimensionality and budget, modDE experiences a more pronounced transition from individual to interaction effects. In contrast, modCMA maintains a stronger emphasis on individual effects, particularly in higher dimensions, even as the budget increases.

By summing all computed effects (cumulative individual, pairwise, and triplet) we get the total explained variance in the algorithm performance (presented in Table V, the "total" column). From this data, we can conclude that the total explained variance for both problem dimensions, all budgets, and the two investigated frameworks, is over 91%, meaning that the results capture a comprehensive picture of how the modules and their interactions influence the algorithm performance. The rest of the variance in the algorithm performance is due to the complex interaction of more than three modules.

TABLE V: Cumulative fraction of variance (in %) in the algorithm performance explained by the individual, pairwise, and triple interaction effects.

algorithm	dim	budget	individual	pairwise	triple	total
modCMA	5	$100d$	41.63	37.72	16.16	95.51
		$500d$	42.05	38.89	15.94	96.88
		$1500d$	43.88	34.87	16.34	95.09
	30	$100d$	60.28	26.66	10.21	97.15
		$500d$	51.16	30.40	13.70	95.26
		$1500d$	54.31	28.95	12.43	95.69
modDE	5	$100d$	80.88	13.12	4.45	98.45
		$500d$	67.67	20.41	8.52	96.60
		$1500d$	40.26	34.04	18.15	92.45
	30	$100d$	68.76	19.26	8.13	96.15
		$500d$	55.88	26.07	11.99	93.94
		$1500d$	42.05	32.61	16.35	91.01

Next, we perform a more detailed analysis to understand the individual, pairwise, and triplet effects.



(a) dim=5, modCMA (b) dim=5, modDE

(c) dim=30, modCMA (d) dim=30, modDE

Fig. 1: Variance in the algorithm performance explained by the individual effects of the algorithm modules of modCMA and modDE, in problem dimension $d = 5$ and $d = 30$ and for budgets of $100d$, $500d$, and $1500d$ function evaluations.

1) Individual module effects: Figures 1a and 1c illustrate the portion of variance explained in the performance of modCMA by the individual effects of each of its modules, at two different problem dimensions (5 and 30) respectively, across three different budget ($100d$, $500d$, $1500d$). The same information for modDE is presented in Figures 1b and 1d. The y-axis presents the percentage of explained variance in algorithm performance, while the x-axis presents the budget used for evaluating the algorithm performance. The colored lines distinguish between the different modules. The legend is displayed below the plots, for each algorithm correspondingly. A summary of the key findings about how the effects are distributed among the modules is presented through the bullet points below:

modCMA - In the low dimensional case for the lower budgets we can observe that *weights_option* and *mirrored* are the most important modules individually, explaining around

15% of the variance in the algorithm performance each, where for the other the modules the explained variance varies between 1% and 5%. The relative importance of *weights_option* suggests that the recombination procedure which is impacted by this module is rather important early in the search. Upon examining the larger budget of 1500d the individual importance of *elitism* becomes more noticeable, as the algorithm has had time to converge, and potentially get stuck in local optima on the multi-modal functions. The importance of *mirrored* remains relatively consistent, suggesting that its variance-reduction effects are beneficial throughout the search. In the higher dimensional case (d=30), irrespective of the budget allocated for evaluating the algorithm performance, consistent patterns emerge for the modules' importance. The *base_sampler* module accounts for a large fraction of the variance in algorithm performance on its own, while the remaining modules remain negligible. The reason for this is the instability of the Halton sampling procedure which when employed within this version of modCMA, causes biases in the sampling directions when the search dimensionality increases.

modDE - The primary module attributing to the variance in the algorithm performance is *lpsr*. At the lower budgets, it explains a substantial part of the algorithm's performance variance (approximately 60%). As the budget increases, this percentage diminishes to approximately 20% for the largest budget. As noted in [5], the reason for the large impact of *lpsr* on modDE's performance is related to the default setting for population size, which is rather small in the wider context of DE. As such, enabling *lpsr* changes the starting population size to be consistent with the popular L-SHADE variant [19] and leads to significant improvements in performance. For high dimensions, comparable trends are observed mirroring those in the low dimensionality scenario.

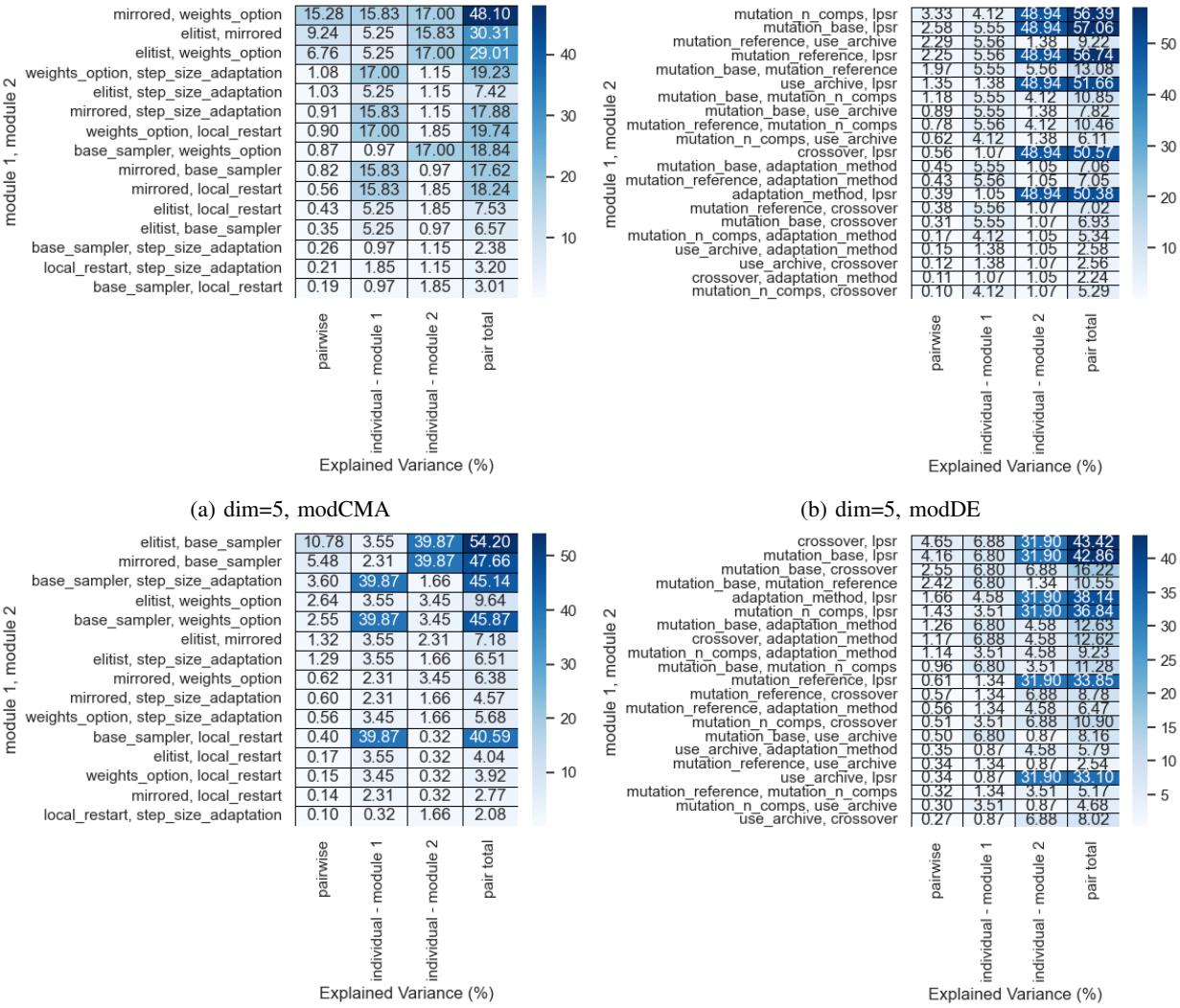
2) Pairwise module effects: Let us assume that we have a pair of modules (Θ_1, Θ_2). The total percentage of variance this pair explains can be calculated as $V_{pair\ total} = V_1 + V_2 + V_{1,2}$. The initial two terms denote the individual importance of the modules, previously examined in the analysis of individual module effects. The last term, the pairwise importance, offers insight into the portion of the explained variance stemming from their interaction effect. The results are presented for one budget (500d) due to the page limit, while for the other budgets, the results are available in our repository.

Figures 2a and 2c depict the percentage of explained variance attributed to each module pair of modCMA for $d = 5$ and $d = 30$, respectively. The heatmap row headers display the module names involved in each pair, while the columns represent: the explained variance resulting from their interaction (the pairwise column), the individual module explained variances (individual - module_1 and individual - module_2 columns), and the total explained variance summing all three (the pair total column). The pairs are sorted in decreasing order based on the pairwise column. The results suggest an interaction between the *mirrored* and *weights_option* module pair for $d = 5$, contributing 15.3% of the overall explained variance. The explained variance of the interaction is almost equal to each individual module's explained variance. Examining the rest of the interactions, we can note that involving the *elitist* module, which by itself is not very important, with either *mirrored* or *weights_option* in combination, their interaction contributes an additional 9.2% and 6.8% in the explained variance, respectively. All other pairwise interactions among the modules fall within similar ranges of explained variance, up to 1%. For $d = 30$, the most impactful pair interaction emerges from the *elitist* and *base_sampler* modules, accounting for 10.8% of the overall explained variance. Next, *mirrored* and *step_size_adaptation* explain a very small fraction of the variance on their own (their individual effect), while there is some additional value when combined with the *base_sampler*. It is noteworthy that pairs involving *base_sampler* yield the highest total explained variance, a result directly influenced by this module's notably high individual explained variance. Meanwhile, the remaining module pairs exhibit contributions ranging from 0% to 3% toward explaining the algorithm's performance variance.

The analyses for modDE are displayed in Figures 2b and 2d for $d = 5$ and $d = 30$, respectively. Across problem dimensions, it is evident that the variance explained by pairs of module interactions ranges between 0% and 3.3% for $d = 5$ and between 0% and 4.6% for $d = 30$. Notably, concerning the total explained variances for both problem dimensions, pairs involving *lpsr* yield higher total explained variances due to this module's notably higher individual explained variance. While the module pairs without *lpsr* explain relatively low amounts of variance overall, it is interesting to note that the combinations with the highest explanatory power are all modules that impact the mutation process. This suggests that picking the right combinations of mutation modules is important in DE, especially in low dimensions. In higher dimensions, the differences between binomial and exponential crossover become more significant, which is also reflected in the fact that combinations involving this module become relatively more impactful in the d=30 case.

3) Triplet module effects: Here the percentage of the variance in the algorithm performance that is explained by the interaction of three modules is presented. Let us assume that we have a triplet of modules $(\Theta_1, \Theta_2, \Theta_3)$. The total percentage of variance a triplet explains can be calculated as $V_{total\ triplet} = V_1 + V_2 + V_3 + V_{1,2} + V_{1,3} + V_{2,3} + V_{1,2,3}$. The terms excluding the last one denote the portion of the explained variance stemming from the modules' individual and pairwise interactions, previously examined in the analysis of the individual and pairwise effects, while the last term is the portion of the explained variance stemming from the triplet interactions of the modules and is investigated next.

Table VI displays the top five triplets, ranked according to the total percentage of explained variance associated with each module triplet. Additionally, it provides the contributions of the triplet interaction ($V_{1,2,3}$) in isolation. The results show that the highest-ranked triplet for modCMA, $d = 5$, and 500d budget consists of *mirrored*, *weights_option*, and *elitist*, which also is the triplet with the highest triplet effect. From the



(a) dim=5, modCMA

(b) dim=5, modDE

(c) dim=30, modCMA

(d) dim=30, modDE

Fig. 2: Variance (%) in the algorithm performance explained by the pairwise effects of the algorithm modules for a,c) modCMA and b,d) modDE in 5 and 30 dimensions respectively, for a budget of 500d.

previous analysis, it is clear that *mirrored* and *weights_option* have relatively large individual effects, meaning they are key drivers of the algorithm's success on their own. Further, their combination with the *elitist* module which is not impactful on its own (refer to individual effects results), adds a lot of value to the total triplet explained variance, visible from the "triplet" column and the pairwise interaction heatmaps in Figure 2a. For higher dimensions, the best triplet is highly influenced by the high individual importance of the *base_sampler* individual importance, while *mirrored* and *elitist* are not so impactful on their own retain their presence in the top triplet through interactions with *base_sampler*.

In the case of modDE, the primary influence on the optimal triplet comes from one module, namely *lpsr*. In lower dimensions, *mutation_base* and *mutation_reference* also play a contributing role through their interaction with *lpsr*. In the higher dimensional case *mutation_base* and *lpsr* remain in the best triplet, while *crossover* replaces the *mutation_reference module*.

B. Problem-level

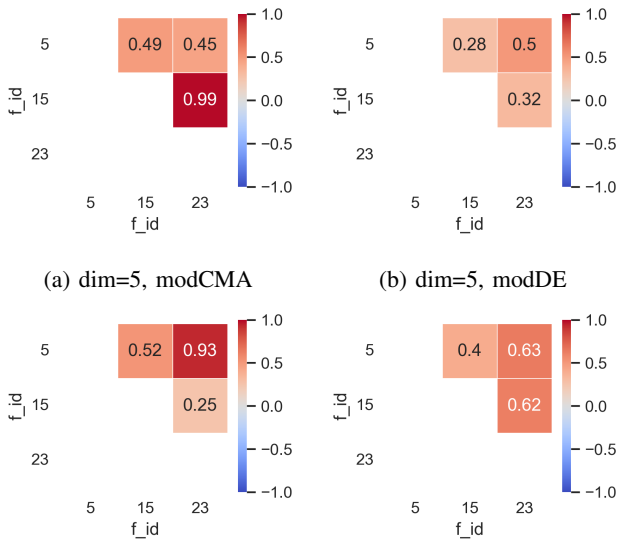
The previous experiment yielded insights into the significance of modules and their interactions in explaining algorithm performance across the entire BBOB benchmark suite. Here, we are showcasing their importance at a problem-specific level. For this purpose, three problems from the BBOB benchmark suite are randomly selected: the 5th (linear slope), 15th (Rastrigin), and 23rd (Katsuuras). Table VII illustrates the combinations of modules with the highest triplet-total effect on the performance, achieved on the 5th, 15th, and 23rd BBOB problems, across dimensions $d = 5$ and $d = 30$ and a budget of 500d. The outcomes are delineated individually for modCMA and modDE. For example, looking into the 5th problem in $d = 30$, we can see that the interaction between the

TABLE VI: Fraction of variance (in %) in the algorithm performance explained by triple interaction effects of the algorithm modules. Only the five module combinations with the largest total importance are displayed.

	d	module1, module 2, module 3	triplet	triplet total
CMA	5	elitist, mirrored, weights_option	8.19	77.54
		mirrored, weights_option, step_size_adaptation	1.19	52.43
		mirrored, weights_option, local_restart	0.95	52.35
		mirrored, base_sampler, weights_option	1.16	51.93
		elitist, mirrored, step_size_adaptation	0.64	34.04
	30	elitist, mirrored, base_sampler	2.96	66.27
		elitist, base_sampler, weights_option	1.71	64.55
		elitist, base_sampler, step_size_adaptation	2.98	63.74
		elitist, base_sampler, local_restart	0.26	55.34
		mirrored, base_sampler, weights_option	1.03	55.31
DE	5	mutation_base, mutation_reference, lpsr	0.89	67.74
		mutation_base, mutation_n_comps, lpsr	0.56	66.25
		mutation_reference, mutation_n_comps, lpsr	0.34	65.31
		mutation_reference, use_archive, lpsr	0.79	62.55
		mutation_base, use_archive, lpsr	0.40	61.08
	30	mutation_base, crossover, lpsr	1.18	58.10
		crossover, adaptation_method, lpsr	0.63	51.46
		mutation_base, adaptation_method, lpsr	0.73	51.09
		mutation_base, mutation_n_comps, lpsr	0.69	49.46
		mutation_n_comps, crossover, lpsr	0.34	49.22

three modules *elitis*, *base_sampler*, and *step_size_adaptation* explains 81.0% of the variance of the algorithm performance achieved on that problem. This variance is a sum of the individual effects (45.2% (*elitist* -M1), 13.7% (*base_sampler* - M2), and 8.0% (*weights_option)* - M3), pairwise effects (5.6% (M1, M3), 3.8% (M1, M2), and 2.9% (M2,M3)), and the triple effect (1.5% (M1, M2, M3)). For the same problem in $d = 5$, it appears that the optimal combination of modules elucidates roughly 30% of the variance, underscoring the critical role of interactions among multiple modules (more than three) in achieving higher explained variances. Additionally, the combination of modules yielding the highest explained variance varies among different problems, which shows that different module interactions are important for solving different problems. We omit to present detailed results for each individual, pairwise, and triplet effects on each problem, however, they are publicly available on our GitHub repository.

Given that we can compute individual (six for modCMA; seven for modDE), pairwise (15 for modCMA; 21 for modDE), and triplet effects (20 for modCMA; 35 for modDE) for each problem-level dataset, we can represent each problem using all quantified effects, 41 in the case of modCMA and 63 for modDE. By analyzing the similarity between problems based on these representations, we can explore which problems exhibit similar module interactions. To show this, we calculate the cosine similarity between the representations of the 5th, 15th, and 23rd problems in the case of modCMA separately for both dimensions (see Figures 3a and 3c) and for modDE (see Figures 3b and 3d). For modCMA in $d = 5$, the module interactions show similarities ($\geq$ 0.9) between the 15th and 23rd problems, whereas for $d = 30$, similar module interactions are observed between the 5th and 23rd problems. For modDE, in $d = 5, 30$, the module interactions provide different patterns for the analyzed problems, achieving similarity up to 0.6.



(a) dim=5, modCMA (b) dim=5, modDE

(c) dim=30, modCMA (d) dim=30, modDE

Fig. 3: Cosine similarity between three BBOB problems represented by the individual, pairwise, and triplet module effects for a,c) modCMA and b,d) modDE for a budget of 500d.

VI. Conclusions

A wide range of modules is available for creating different variants of modular optimization algorithms designed for addressing single-objective black-box numerical optimization problems, but knowledge about the individual impact of each module on the algorithm's effectiveness, as well as how these modules interplay when integrated, is relatively insufficient. We employed the f-ANOVA framework [9] to investigate the influence of modules and their interactions. Our focus was on two modular frameworks, namely modular Covariance Matrix Adaptation - Evolutionary Strategy (modCMA) and modular Differential Evolution (modDE), applied to 24 problems from the BBOB benchmark suite in dimensions 5 and 30. With 324 variants for modCMA and 576 for modDE, our goal was to assess individual and interaction module effects in two distinct scenarios: one evaluating the modules' impact on solving the entire benchmark suite (encompassing all problem instances), and the other examining their influence on a problem-specific level.

We have shown that for a problem dimension of 5 and modCMA, *weights_option* and *mirrorred* play a substantial role in explaining most of the variance, albeit not significantly larger than other modules. Conversely, in the context of a problem dimension of 30, *base_sampler* exhibits the highest individual effect, contributing to half of the explained variance in algorithm performance. For modDE, *lpsr* emerges as the primary factor explaining most of the variance for both problem dimensions. When comparing modCMA and modDE, both algorithms exhibit changes in the importance of individual modules and their interactions with variations

TABLE VII: The combinations of modules with the largest total importance for the 5th, 15th, and 23rd BBOB problems in $d = 5$ and $d = 30$ for a budget of $500d$. The results are presented separately for modCMA and modDE.

		modCMA		modDE	
dim	f_id	module 1, module 2, module 3	triplet-total	module 1, module 2, module 3	triplet-total
5	5	elitist, weights_option, step_size_adaptation	29.52	mutation_base, mutation_n_comps, use_archive	23.73
	15	elitist, base_sampler, local_restart	62.4	mutation_base, mutation_reference, lpsr	67.44
	23	elitist, mirrored, weights_option	78.99	mutation_reference, use_archive, lpsr	39.03
30	5	elitist, base_sampler, weights_option	81.08	mutation_base, mutation_reference, lpsr	46.07
	15	elitist, mirrored, base_sampler	69.93	crossover, adaptation_method, lpsr	57.71
	23	elitist, mirrored, step_size_adaptation	84.73	mutation_base, crossover, lpsr	57.59

in dimensionality and budget. Notably, modDE experiences a more noticeable transition from individual to interaction effects, whereas modCMA follows the opposite trend. The problem-level findings suggest a positive indication for identifying problems where module interactions contribute equally to algorithm performance.

Some limitations of the approach are that the calculation of module interactions requires an exponential amount of time, thus we are not able to calculate interactions beyond three modules in a reasonable amount of time.

A practical use case of the proposed approach can be the post-hoc analysis of algorithm configuration (AC) where the most important algorithm parameters are identified. Another example is the reduction of the module space as we can select the sub-space of the most influential algorithm modules and perform AC, which may lead to more promising results.

In future work, we plan to investigate another modular framework, PSO-X [6]. We also aim to identify problems exhibiting analogous modular interactions and establish connections between these findings and the inherent landscape properties of the problems under consideration together with the algorithm behavior (i.e., trajectories). This approach will offer additional assistance to designers of modular frameworks, providing them with deeper insights into the specific modules and the nature of interactions required for addressing certain landscape properties. We are also planning to test another approach for assessing module importance, the Pearson divergence ANOVA (PED-ANOVA) to calculate cross-form hyperparameter importance in arbitrary spaces [20].

REFERENCES

[1] Á. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed., ser. Natural Computing. Springer, 2015.

[2] J. Stork, A. E. Eiben, and T. Bartz-Beielstein, "A new taxonomy of global optimization algorithms," *Natural Computing*, vol. 21, no. 2, pp. 219–242, 2022. [Online]. Available: https://doi.org/10.1007/s11047-020-09820-4

[3] J. Dreo, A. Liefooghe, S. Verel, M. Schoenauer, J. J. Merelo, A. Quemy, B. Bouvier, and J. Gmys, "Paradiseo: from a modular framework for evolutionary computation to the automated design of metaheuristics: 22 years of paradiseo," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021, pp. 1522–1530.

[4] 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.

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

[6] 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.

[7] N. Hansen, A. Auger, and D. Brockhoff, "Data from the BBOB workshops," https://numbbo.github.io/data-archive/bbob/, 2020.

[8] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[9] 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.

[10] A. Nikolikj. (2024) Footprints mtr. [Online]. Available: https://github.com/anikolik/fanova.git

[11] 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.

[12] 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.

[13] G. Hooker, "Generalized functional anova diagnostics for high-dimensional functions of dependent variables," *Journal of Computational and Graphical Statistics*, vol. 16, no. 3, pp. 709–732, 2007.

[14] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[15] A. Kostovska, D. Vermetten, S. Džeroski, P. Panov, T. Eftimov, and 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.

[16] J. de Nobel, F. Ye, D. Vermetten, H. Wang, C. Doerr, and T. Bäck, "IOHexperimenter: benchmarking platform for iterative optimization heuristics," *CoRR*, vol. abs/2111.04077, 2021. [Online]. Available: https://arxiv.org/abs/2111.04077

[17] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions," INRIA, Tech. Rep. RR-6829, 2009. [Online]. Available: https://hal.inria.fr/inria-00362633/document

[18] 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, pp. 114–144, 2020.

[19] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE congress on evolutionary computation*. IEEE, 2013, pp. 71–78.

[20] S. Watanabe, A. Bansal, and F. Hutter, "PED-ANOVA: efficiently quantifying hyperparameter importance in arbitrary subspaces," in *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI-23*, 2023, pp. 4389–4396.