Algorithm Configuration Data Mining for CMA Evolution Strategies

Sander van Rijn
LIACS, Leiden University
Niels Bohrweg 1
Leiden, the Netherlands
s.j.van.rijn@liacs.leidenuniv.nl

Hao Wang
LIACS, Leiden University
Niels Bohrweg 1
Leiden, the Netherlands
h.wang@liacs.leidenuniv.nl

Bas van Stein
LIACS, Leiden University
Niels Bohrweg 1
Leiden, the Netherlands
b.van.stein@liacs.leidenuniv.nl

Thomas Bäck
LIACS, Leiden University
Niels Bohrweg 1
Leiden, the Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

ABSTRACT

In the past years, quite a number of algorithmic extensions of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) have been proposed. These extensions define a large algorithm design space, but relatively little is known about the performance of most of these variations and the interaction between them.

In this paper we investigate how various algorithmic extensions interact and what their impact is on objective functions from the Black Box Optimization Benchmark (BBOB). Based on the existing Estimated Running Time (ERT) and Fixed Cost Error (FCE) measures, a novel algorithm quality measure is proposed to quantify an impact-score of the variants studied.

Using performance data from running 4,608 available algorithmic variations in the configurable CMA-ES framework published previously, decision trees and other data mining methods are used to analyze performance data. Analysis identifies algorithmic variations required for obtaining best performance and identifies strong differences between objective functions, thereby helping to understand the interaction of algorithmic components for an objective function and, ultimately, for an objective function class. The results also quantitatively confirm that popular variants such as increasing population size and elitism generally have a positive impact on algorithm performance.

CCS CONCEPTS

•**Theory of computation** → **Evolutionary algorithms;** *Algorithm design techniques; Theory of randomized search heuristics;*
•**Mathematics of computing** → *Exploratory data analysis;*

GECCO '17, Berlin, Germany

978-1-4503-4920-8/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3071178.3071205

KEYWORDS

Evolution Strategies, Metaheuristics, Parameter tuning, Empirical study, Performance measures

ACM Reference format:
Sander van Rijn, Hao Wang, Bas van Stein, and Thomas Bäck. 2017. Algorithm Configuration Data Mining for CMA Evolution Strategies. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071205

1 INTRODUCTION

Heuristic optimization methods have long been a topic of study, and have found their way into many industrial applications. Evolutionary algorithms such as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10] are among the most well-known and widely used of these methods. For most optimization problems faced in practice, some default optimizer with at least reasonable performance is usually available. As the use of optimization for computationally expensive problems continues to grow, running times grow accordingly. To mitigate this as much as possible, it is important to use the best performing optimizer. This *Algorithm Selection Problem* [15] has led to a lot of research into trying to find better algorithms.

Among these algorithms, many variations on the CMA-ES, such as Active Update [12], Mirrored (Orthogonal) Sampling [1, 17] and (B)IPOP [2, 8], have been proposed during recent years. Each of these aims to improve performance in general or for specific function landscapes. New variants are still regularly introduced, but their synthesis is complex and involves a lot of human expertise, while at the same time the interactions of many of the existing variations are not sufficiently well understood yet. Exploratory landscape analysis [4, 13] can steer this creation process in interesting directions by identifying landscape properties and how they interact with certain algorithmic techniques and variants of the CMA-ES.

Beside different variants, many algorithms also have strategy parameters that can be tuned to optimize performance for specific cases. Automated algorithm configuration methods such as SMAC [11] have recently gained popularity for such optimizer-tuning problems. These methods provide useful improvements for each

particular problem instance, but little general knowledge is usually gained about the performance properties of the *algorithm*.

In this paper we present a data mining approach on performance data for a large number of CMA-ES algorithm configurations, created using the modular framework introduced by van Rijn *et al.*[16]. This approach consists of introducing a mapping to combine the Estimated Running Time (ERT) and Fixed Cost Error (FCE) measures into a single value, which can be used to calculate a quantative *impact score* for each variant in the configuration space. Furthermore, the performance ranking of all these configurations, and more specifically the distribution of modules across this ranking, is used in an attempt to gain insight into properties of algorithms. Eventually, these properties should be linked to the landscape properties already identified by earlier research.

The rest of this paper is organised as follows. Details of the algorithm configuration data used in this research are presented in Section 2. This includes a short description of the modular CMA-ES framework used to generate it. Our proposed quality measure and impact score comparison method are defined in Section 3. Section 4 details the various analyses performed using decision trees (Section 4.1), impact score (Section 4.2) and relative activation correlation (Section 4.3). Finally, an overview of our conclusions and some suggestions for future work are presented in Section 5.

2 DATASET

In this research we use algorithm benchmarking results obtained using the modular CMA-ES framework by van Rijn *et al.*[16] in which eleven variants are encoded. Referred to as *modules*, these can be activated independently and combined arbitrarily. A list of all modules and their available options are listed in Table 1.

Runs for all 24 noiseless functions of the BBOB suite [9] are performed in $D = 2, 3, 5, 10, 15, \ldots, 35, 40$ dimensions. Each of the $24 \cdot 10 = 240$ (function, dimensionality) combinations will be referred to as an *experiment*. Results for one experiment contain data records for all possible $2^9 \times 3^2 = 4,608$ CMA-ES configurations.

Each data record $(\mathbf{c}_i, e_i, f_i)$ consists of a *configuration* specification $\mathbf{c}_i \in \{0,1\}^9 \times \{0,1,2\}^2$ and a pair of quality measures: the *Estimated Running Time* (ERT) $e_i = \mathrm{ERT}(\mathbf{c}_i) \in \mathbb{R}$ and *Fixed Cost Error* (FCE) $f_i = \mathrm{FCE}(\mathbf{c}_i) \in \mathbb{R}$. The values for e_i and f_i are calculated from 32 independent runs per configuration $\mathbf{c}_i$, as suggested in [16]. Each run is performed using an evaluation budget $b = 1\,000 \cdot D$.

3 QUANTITATIVE PERFORMANCE MEASURE

Quantitative comparisons between optimization algorithms in benchmarking environments are ideally based on the required runtime. The ERT score introduced by Auger and Hansen [2] provides such a numerical measure based on the expected number of function evaluations required to reach a predefined target value. In practice, this means ERT scores cannot always be measured when the allowed evaluation budget is not sufficient to reach the target value. Only a qualitative comparison can be made using the FCE score in these cases. We propose a quality measure that combines the ERT and FCE values into a single value.

This use of two different quality measures currently poses a problem when performing quantitative analysis on the performance data described in Section 2. Both quality measures cannot be compared to each other and are not directly comparable between different benchmark functions. As a first step around this problem, an algorithm's FCE is only considered relevant in absence of an ERT score. In other words, any algorithm with an ERT score is defined to always be superior to an algorithm without it. This follows the existing convention in reporting on BBOB results, and has already been proposed in [6].

Table 1: *Overview of the available ES modules in the modular CMA-ES framework from [16].* For most of these modules the only required options are *off* and *on*, encoded by the values 0 and 1. For quasi-Gaussian sampling and increasing population, the additional option is encoded by the value 2. The entries in row 9, recombination weights, specify the formula for calculating each weight w_i.

#	Module name	0 (default)	1	2
1	Active Update [12]	off	on	-
2	Elitism	(μ, λ)	$(\mu+\lambda)$	-
3	Mirrored Sampling [5]	off	on	-
4	Orthogonal Sampling [17]	off	on	-
5	Sequential Selection [5]	off	on	-
6	Threshold Convergence [14]	off	on	-
7	TPA [7]	off	on	-
8	Pairwise Selection [1]	off	on	-
9	Recombination Weights	$\frac{\log(\mu+\frac{1}{2})-\log(i)}{\sum_j w_j}$	$\frac{1}{\mu}$	-
10	Quasi-Gaussian Sampling [3]	off	Sobol	Halton
11	Increasing Population [2, 8]	off	IPOP	BIPOP

Next, a useful normalization must be defined to allow comparisons between functions. By their definition, normalizing ERT values is straight-forward. As a doubling of an algorithm's ERT indicates a double expected running time, these values can be normalized linearly. The minimum ERT value is always greater than zero, and the maximum is defined as

$$\mathrm{ERT}_{\max} = k \cdot b,$$

where k is the number of runs and b the evaluation budget. Normalization can therefore be applied through division by $\mathrm{ERT}_{\max}$.

FCE values are typically plotted on a log-scale in convergence analysis, suggesting that normalization is best done after taking the logarithm. Normalizing this range is still not trivial. The maximum value for most fitness functions is either undefined or bound by the input domain. $\mathrm{FCE}_{\max}$ is therefore most easily fixed to the maximum value found in all k runs. However, the minimum value can clearly be set to the target value used for ERT calculations. After all, an ERT score can be calculated for any algorithm that reaches a lower FCE value.

Combining the above, a single value q can be defined for configuration $\mathbf{c}$ from a given set of algorithm configurations C, as follows:

$$q(\mathbf{c}) = \begin{cases} \dfrac{\mathrm{ERT}(\mathbf{c})}{\mathrm{ERT}_{\max}} & \text{if } \mathrm{ERT}(\mathbf{c}) \text{ exists} \\ 1 + \dfrac{\log(\mathrm{FCE}(\mathbf{c})/\mathrm{FCE}_{\mathrm{target}})}{\log(\mathrm{FCE}_{\max}/\mathrm{FCE}_{\mathrm{target}})} & \text{otherwise,} \end{cases}$$

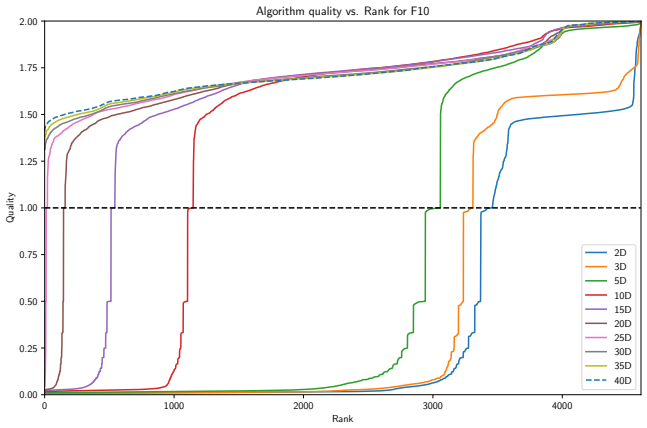

Figure 1: *Quality Measure vs Rank.* The above graph shows the proposed quality score for the $4,608$ algorithm configurations, sorted according to their quality q on the BBOB F10 function in $2, 3, 5, 10, 15, \ldots, 35, 40$ dimensions.

where $\mathrm{FCE}_{\max} = \max\{\mathrm{FCE}(\mathbf{c}) \mid \mathbf{c} \in C\}$, and $\mathrm{FCE}_{\mathrm{target}}$ is the FCE value used as a target for calculating the ERT.

This maps the (ERT, FCE) pair to the range $[0, 2]$ and allows for a quantitative comparison between various algorithms and fitness functions. An example of the quality score progression from the dataset can be seen in Figure 1. By choosing the range to be $[0, 2]$, the data can now be used for computational analysis. For $q < 1$, our new q-measure always provides an ERT score.

3.1 Impact Score

The quality measure q proposed above is designed for configurations instead of modules. In order to investigate the *impact* of each module and the combination of modules, we propose the following *impact score* definition.

Divide the given set of configurations C into the desired subsets for comparison. To analyze a module x, we obtain subsets $\mathrm{C}^x_{\mathrm{on}}$ and $\mathrm{C}^x_{\mathrm{off}}$ of configurations where module x is *on* or *off* respectively. Let

$$q(\mathrm{C}) = \{q(\mathbf{c}) \mid \mathbf{c} \in \mathrm{C}\}$$

be the set of quality values q associated with configuration set C.

We expect the distribution of $q(\mathrm{C}^x_{\mathrm{off}})$ and $q(\mathrm{C}^x_{\mathrm{on}})$ to differ from each other if some module x has a significant impact on algorithm performance. However, if module x has no impact, the configurations in $\mathrm{C}^x_{\mathrm{on}}$ and $\mathrm{C}^x_{\mathrm{off}}$ are expected to be randomly interleaved when sorted by performance q, resulting in a very similar distribution. The quantitative impact score I is defined as the difference between the mean quality scores $\bar{q}(\mathrm{C})$ of the subsets to be compared:

$$I_x = \bar{q}(\mathrm{C}^x_{\mathrm{off}}) - \bar{q}(\mathrm{C}^x_{\mathrm{on}}),$$

where the mean quality value $\bar{q}(\mathrm{C})$ is defined as follows:

$$\bar{q}(\mathrm{C}) = \frac{\sum_{i=0}^{|\mathrm{C}|} q(\mathbf{c}_i)}{|\,\mathrm{C}\,|}.$$

Assuming minimization, a positive I_x score indicates a positive impact of module x, and a negative I_x indicates a negative influence.

Table 2: *Feature importance score per module.* Calculation of this score is performed by generating 250 randomized decision trees on all $4,608$ configurations, limited to at least 20 configurations per leaf node. Listed values are averaged over the results of all 240 experiments.

Module name	Feature importance
Active	0.071
Elitism	**0.197**
Mirrored	0.018
Orthogonal	0.021
Sequential Selection	0.026
Threshold	**0.307**
TPA	**0.163**
Pairwise Selection	0.053
Weights	0.043
Base-Sampler	0.056
(B)IPOP	0.044

Because no assumption of normality can be made on the distrbution $q(\mathrm{C})$, the two-tailed nonparametric *Mann-Whitney U* statistical test is adopted to determine the significance of impact score I_x.

4 ANALYSIS

This section describes the various methods used to gain insight in the performance of different CMA-ES variants. First the use of decision trees is described in Section 4.1. Next, Section 4.2 explores the impact as defined in Section 3.1, both on a per-module and module interaction basis. Finally Section 4.3 examines how the module activation is distributed across configurations. The dataset and a *Python Notebook* documenting the procedures are available online on GitHub[1].

4.1 Decision Tree Feature Importance

Decision trees are commonly used in exploratory data mining, because they capture the features that most prominently define subgroups of the data. In this context of algorithm configurations, we expect decision trees to be useful. After all, we ultimately wish to be able to create decision trees based on function properties to select modules that will benefit the optimization process. Although this research does not yet make the step towards identifying properties of the various fitness landscapes, some empirical decision trees can be created for each experiment.

The decision trees for each experiment perform selection according to the quality measure q previously defined in Section 3. Because eleven modules can be used to split nodes, the entire trees quickly become too large to be human-readable. However, they still contain useful information in terms of which module is selected for at which point in the tree, also known as the *feature importance* measure. Table 2 lists values for each module, calculated as the mean over all 240 experiments.

[1] https://github.com/Energya/cma-es-configuration-data-mining

Table 3: *Impact score and statistical significance per module.* The values shown are calculated using aggregated performance data on all 4,608 configurations for each of the 240 experiments. P-values are calculated using the two-tailed *Mann-Whitney U* test.

Module name	I_{module}	p-value
Active	-0.111	0.0
Elitism	**0.165**	**0.0**
Mirrored	0.013	$7.65 \cdot 10^{-84}$
Orthogonal	0.040	0.0
Sequential Selection	-0.070	0.0
Threshold	**-0.398**	**0.0**
TPA	0.072	0.0
Pairwise Selection	0.004	$4.25 \cdot 10^{-18}$
Weights	-0.091	0.0
Base-Sampler	-0.002	$1.52 \cdot 10^{-21}$
(B)IPOP	0.061	0.0

A high value for e.g. Threshold convergence is an indication that a particular module is often used for selection near the top of the trees. Other modules that score high in terms of feature importance are Elitism and TPA. The lower values for Mirrored, Orthogonal and Base-Sampler show that selection according to these modules does not provide much information to the decision process.

The feature importance values indicate how well a given module can be used to split the data, but not whether this module should be activated in general to obtain better results. For this, individual trees would have to be examined, or a single tree has to be created from the results of all experiments. Instead, we continue the analysis using the *impact scores* defined in Section 3.1.

4.2 Impact Scores

First we examine impact scores based on the activation of single modules, followed by a discussion of module interactions.

4.2.1 Single Module. Table 3 shows the impact scores and p-values of all modules, aggregated over all 240 experiments. These results exhibit a clear difference in impact score between the various modules. Elitism has the most positive impact score of 0.165, while Threshold convergence is the most negative at −0.398. Other modules such as Pairwise Selection seem to have no overall impact, with a negligible score of 0.004.

Exploring further, impact scores for various modules are examined for each experiment separately. Figure 2 shows impact score heatmaps by objective function for three example modules: Threshold, Orthogonal and Elitism. The overall impact scores from Table 3 are clearly reflected in the figures, but differences in behavior for the various experiments are also clearly visible.

Patterns can be seen both in the functions and along the dimensionality. For example, Threshold performs increasingly worse in higher dimensions for the F1, F5 and F21 functions in particular, while improving in performance in higher dimensionalities

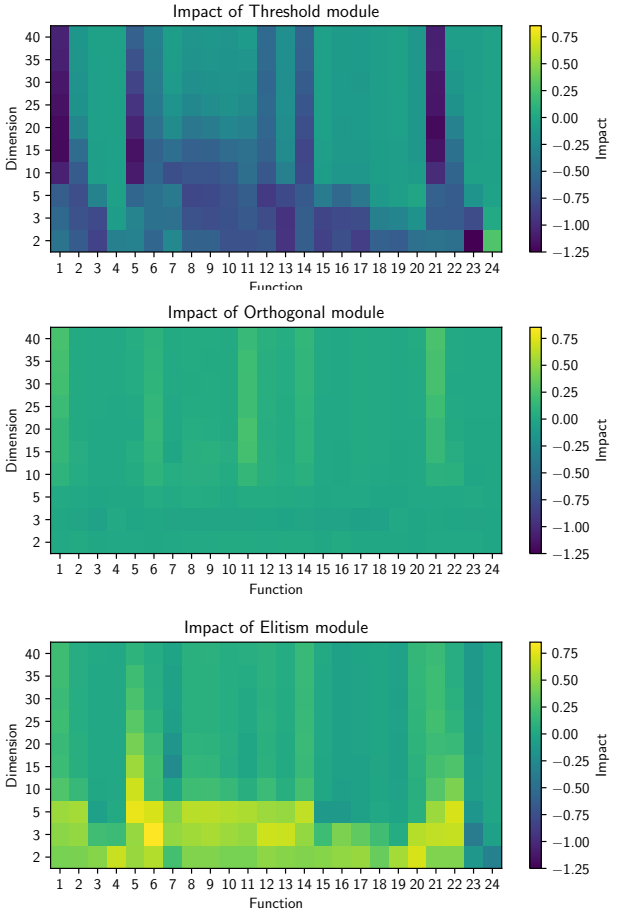

Figure 2: *Heatmaps of module impact per experiment.* The heatmaps show impact values for the Threshold convergence (top), Orthogonal Sampling (middle) and Elitism (bottom) modules. All impact scores outside the range $[-0.144, 0.229]$ are statistically significant according to the two-tailed Mann-Whitney U test ($P < 10^{-4}$). P-values for the remaining impact scores can be greater than 10^{-4} and should be examined on a case-by-case basis.

for other functions such as F3, F13 and F23. Meanwhile, Elitism performs exceptionally well in lower dimensionalities, but loses its positive impact in experiments with more than five dimensions. Orthogonal however slightly increases performance in higher dimensionalities. The experiments in $10–40D$ seem to show a constant pattern for all modules towards higher dimensionalities.

4.2.2 Module Interaction. In addition to investigating the impact of single modules, the same can be done for the interactions between modules. To calculate the impact of two modules x and y when activated together, the set of configurations C is divided into subsets C^{xy}_{on} and C^{xy}_{off}. The *on* subset C^{xy}_{on} consists of all configurations in which both module x and y are selected. The *off* subset is then constructed as $C^{xy}_{\text{off}} = C - C^{xy}_{\text{on}}$.

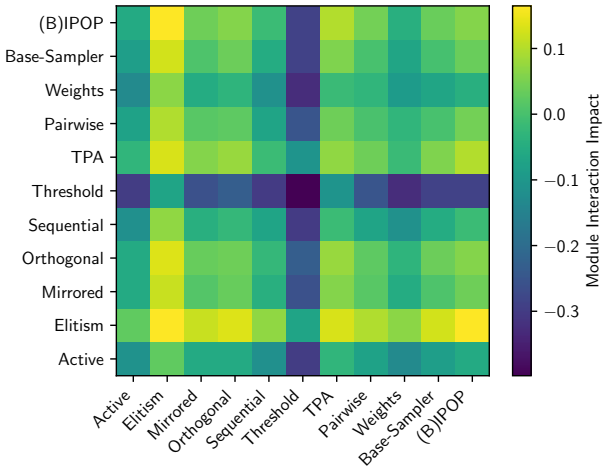

Figure 3: *Aggregate Module Interaction Impact.* Displayed impact scores are calculated as a mean over all 240 experiments. Each score indicates the difference in q-score between the set of configurations with two modules x and y active, versus the remaining configurations. The heatmap is symmetrical along the diagonal $x = y$, which indicates the single module impact as shown in Table 3.

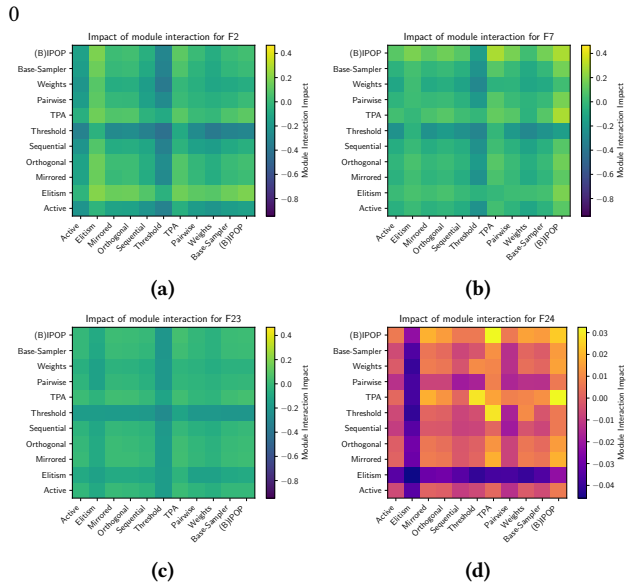

Figure 4: *Heatmaps of module interaction impact per function.* Displayed impact scores are averaged over the experiments in 10 different dimensionalities per function. Note that Figure 4d uses a different scale than the other three heatmaps because of the smaller range in values.

Figure 3 shows a heatmap of the mean impact scores for each module combination, averaged over all 240 experiments. Again, the overall positive and negative impact of Elitism and Threshold convergence can be clearly identified. It is also interesting that

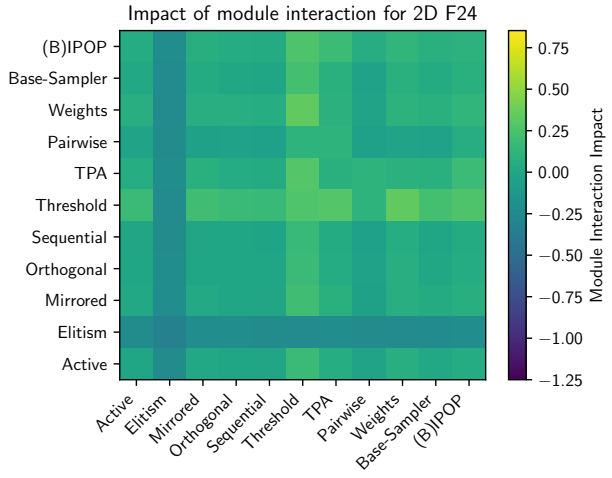

Figure 5: *Module Interaction Impact Score heatmap for 2D F24.* Note the different value range compared to Figure 3.

the positive and negative impact of these two modules cancel each other out when they are combined.

A known combination that is also interesting to look at, is the addition of Pairwise Selection to Mirrored Sampling. Having been added to maintain the properties of Mirrored Sampling while avoiding a possible step-size bias, one would expect a (more) positive impact of their combination than both separate and random. However, Figure 3 shows their combined impact to be effectively neutral.

Separating the impact scores by function already shows more diversity. Four examples are shown in Figure 4. Most heatmaps resemble that of the global distribution, as can be seen in Figure 4a: Elitism and Threshold convergence are most prominent in terms of positive and negative impact respectively. An interesting observation for the F2 function is the combination of Elitism and (B)IPOP, providing the highest average positive impact over all tested dimensionalities.

Impact heatmaps for other benchmark functions highlight other interesting combinations of modules. Effective interactions between Elitism, TPA and (B)IPOP can be seen in Figure 4b, while Elitism in particular is best avoided for F23 as can be seen in Figure 4c.

A majority of interactions do not seem to produce any significant impact at all, of which the heatmap in Figure 4d for F24 is a good example. This apparent lack of significant interactions still contains useful information: Threshold convergence and Elitism also do not stand out in terms of impact, suggesting a different performance for this function than for others. The best performing combinations for these experiments actually seem to be the combination of Threshold convergence with TPA and/or (B)IPOP. Examining the heatmap Figure 5 for 2D F24 in particular confirms this with greater absolute impact values.

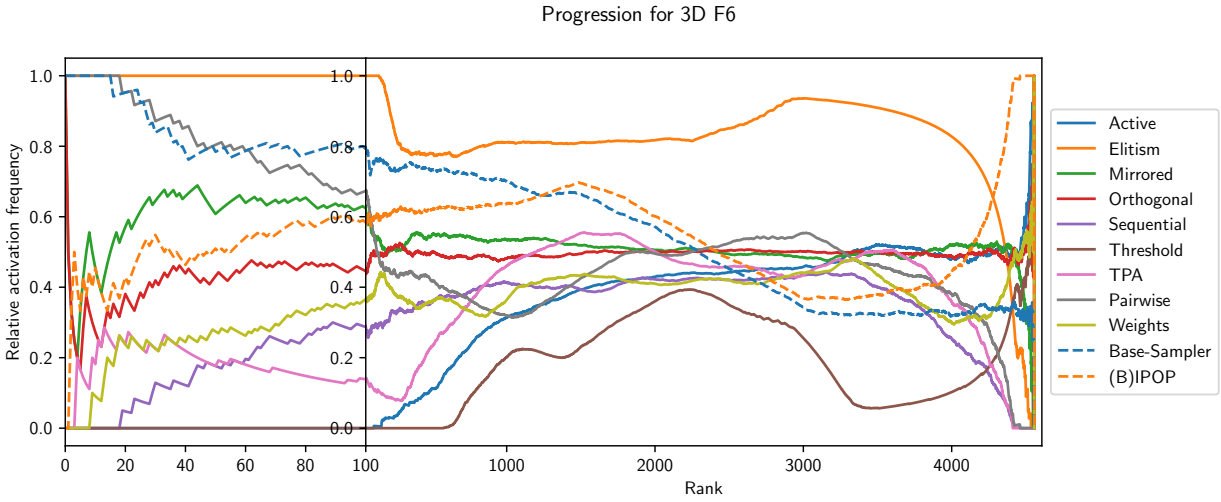

Figure 6: *Relative Module Activation.* The above graph shows the fractional activation of each module for the 4 608 algorithm configurations, sorted by their quality on the BBOB F6 function in 3 dimensions. For rank n, the value for a module x is the number of configurations in which this module is active in the top n ranked configurations, divided by n. A value of 1 indicates it has always been chosen up to that point, while 0 shows the opposite. The value for alternatingly active modules trends to 0.5.

4.3 Relative Module Activation

The impact score I_x as used above gives a quantitative indication of the difference in quality distributions, but the distribution itself can also be examined. If a module has a positive impact on performance, it will be activated more frequently in configurations that are ranked higher. By visualizing the relative activation frequency of a module in the top n configurations, an indication of a module's *behavior* through the sorted configuration space is obtained.

The example in Figure 6 shows a high activation frequency of several modules in the top-ranked configurations for a particular function. Elitism is seen to be selected in all top-100 ranked configurations, while Active and Threshold are avoided for at least the top-100 ranked cases. This information is reflected by the impact score heatmaps in Figure 2: Elitism scores high, Threshold scores low and the neutral impact score of Orthogonal corresponds to the relative selection frequency of 0.5.

4.3.1 Correlation Clustering. In the relative module activation plots, we can search for similar behavior of different modules. A numerical comparison of these behaviors is possible by calculating the Pearson correlation between the various progressions. In total, behavior for all eleven modules in each of the 240 experiments is known, for a total of 2, 640 behaviors.

Displaying the correlation values for all $2,640 \times 2,640$ makes no sense for two reasons. First, this would result in an extremely large visualization that is no longer human-readable. Second, many correlations would be between the behavior of two different modules in different experiments. Such correlation values have little or no meaning.

Figure 7 shows correlation heatmaps between all experiments where the module has been fixed to Elitism or Threshold. Most clear is the clustering by dimensionality: 2 and $3D$ in the bottom-left, $5, 10, \ldots, 35, 40D$ in the top-right. Similar clustering can be seen in the heatmaps of all other modules, but this is too coarse-grained to produce useful conclusions.

By instead selecting only clusters of behaviors with the highest correlation values per module, an indication of *experiment similarity* can be obtained. This is especially true when particular modules perform very well.

Three best examples of highly clustered behaviors are shown in Figure 8. These show the clustering of Elitism and Threshold modules for the similar Ellipsoidal and Discus functions (F2, F10 and F11) in 2D, and of the Step Ellipsoidal function (F7) in $10 - 40D$. All modules are active almost all of the top-100 ranking configurations.

Besides identifying experiment similarity, behavior correlation can also be used to identify successful cooperation between modules. If two modules *cooperate* in a particular experiment, their selection progression can be expected to correlate with each other as well.

As can be expected, most modules do not correlate very well with each other. Of the combinations that do correlate, the majority show exceptionally bad rather than good performance. The two most interesting cases with both high correlation and a positive impact are shown in Figure 9.

5 CONCLUSIONS AND OUTLOOK

In this paper, the performance of various CMA-ES extensions is investigated over the BBOB suite functions, using the results previously obtained from a huge variety of CMA-ES configurations. Furthermore, a novel quality measure (q) is proposed to quantify the impact score of a given CMA-ES configuration by combining the well-known ERT and FCE measures. This score enables the quantitative comparison between configurations.

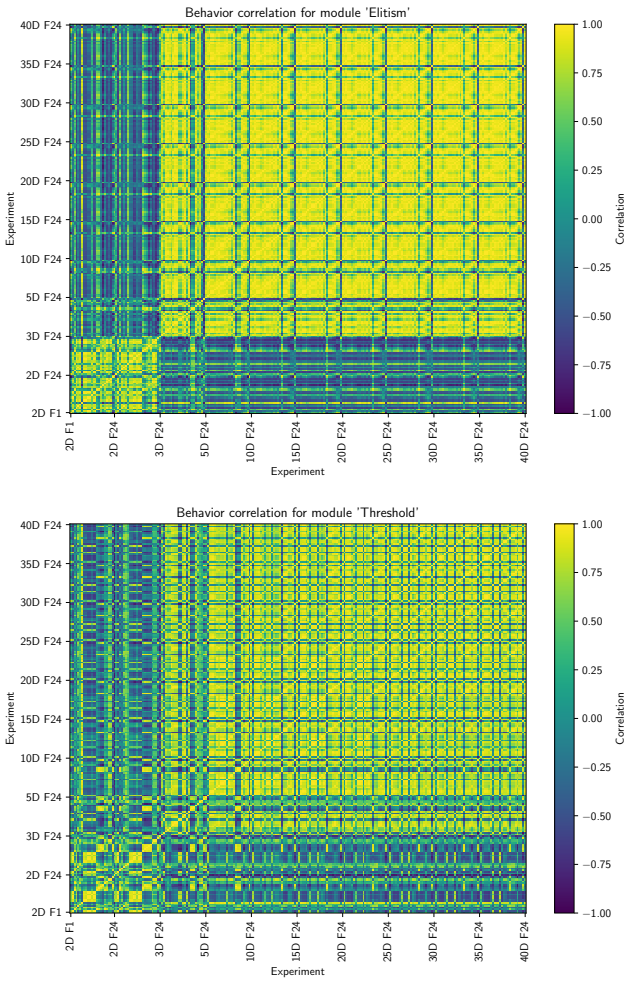

Figure 7: *Correlation of Module Selection Progression.* Correlation of the "behavior" of the Elitism (top) and Threshold convergence (bottom) module between the 240 different experiments.

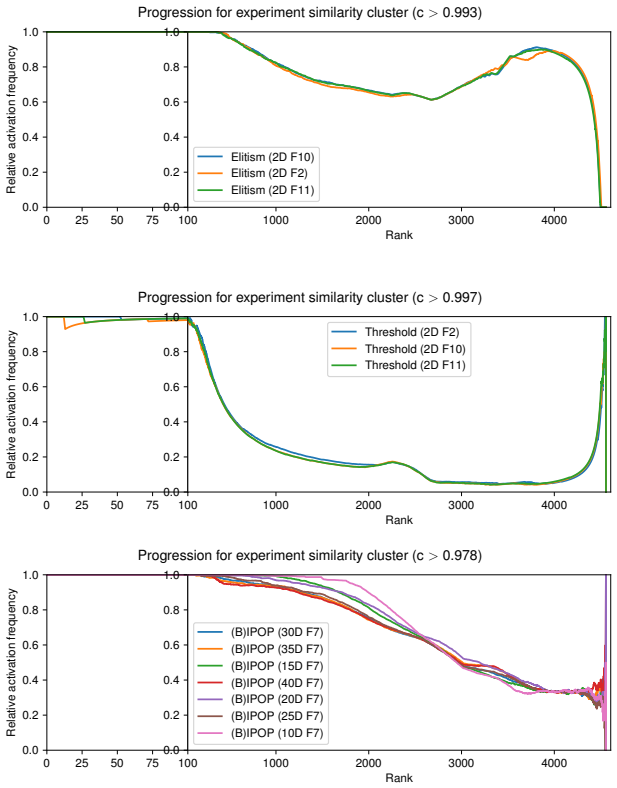

Figure 8: *Three clusters of highly correlated Module Selection Progressions.* Plots of the "behavior" of the Elitism (top), Threshold convergence (middle) and (B)IPOP module (bottom). For Elitism and Threshold convergence, F2, F10 and F11 in 2D are shown, while F7 is shown in 10 – 40D for (B)IPOP.

Additionaly, a generic methodology for mining and analyzing optimization algorithm features and the mapping to function features is proposed and applied. Conclusions of analyses through decision tree feature importance, impact scores and module behavior correlation confirm each other. This indicates their potential use for further research into mining algorithm configurations.

Through this analysis, an overall impacts for several CMA-ES variants are shown on the BBOB suite: positive for Elitism and (B)IPOP, and negative for Threshold convergence. The magnitude of these impact scores corresponds highly to the standard feature importance measure as determined by creating decision trees. Additionally, impact scores for module interactions can be used to identify interesting combinations of variants for specific functions.

As suggested in Section 4.2, experiments in higher dimensions on the BBOB suite are required to provide more information on some of the dimension-related trends that seem to be present. Furthermore, applying the presented methods to a larger set of algorithm configurations would increase the available knowledge about the effects of different algorithm variations. This can be done by expanding the set of available variations in the framework used to generate the data for this experiment, or by adapting these methods to work with other existing frameworks.

Additionally, this research can be linked to earlier studies on determining properties of fitness landscapes. By collecting a large body of algorithm performance knowledge on various function landscapes, steps can be made towards identifying properties related to the studied algorithm variants and creating a mapping between properties of algorithms and fitness functions.

REFERENCES

[1] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. 2011. Mirrored sampling in evolution strategies with weighted recombination. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '11)*. ACM, 861–868.

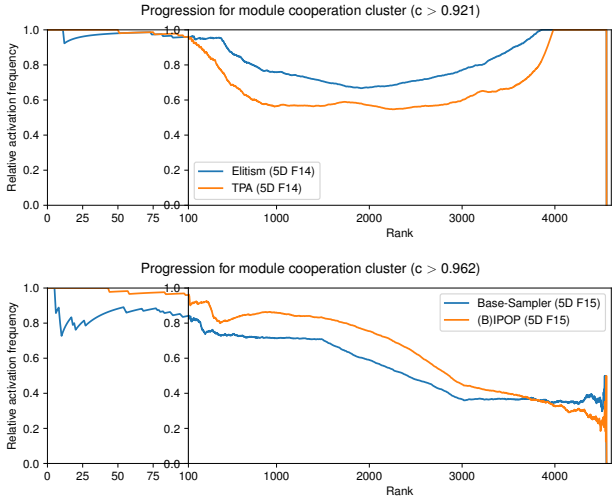

Figure 9: *Two clusters of highly correlated Module Selection Progressions.* Plots of the Elitism and TPA modules for F14 in 5D (top) and (B)IPOP and Base-Sampler modules in 5D F15.

[2] Anne Auger and Nikolaus Hansen. 2005. A restart CMA evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Vol. 2. IEEE, 1769–1776.

[3] Anne Auger, Mohammed Jebalia, and Olivier Teytaud. 2006. Algorithms (x, sigma, eta): quasi-random mutations for evolution strategies. In *Artificial Evolution: 7th International Conference, Revised Selected Papers*. Springer, 296–307.

[4] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuß. 2012. Algorithm Selection Based on Exploratory Landscape Analysis and Cost-sensitive Learning. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO '12)*. ACM, New York, NY, USA, 313–320. DOI: http://dx.doi.org/10.1145/2330163.2330209

[5] Dimo Brockhoff, Anne Auger, Nikolaus Hansen, Dirk V. Arnold, and Tim Hohm. 2010. Mirrored Sampling and Sequential Selection for Evolution Strategies. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*, Robert Schaefer, Carlos Cotta, Joanna Ko lodziej, and Günter Rudolph (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 11–21. DOI: http://dx.doi.org/10.1007/978-3-642-15844-5_2

[6] Pilar Caamaño, Jose A. Becerra, Francisco Bellas, and Richard J. Duro. 2011. Are Evolutionary Algorithm Competitions Characterizing Landscapes Appropriately. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '11)*. ACM, New York, NY, USA, 695–702. DOI: http://dx.doi.org/10.1145/2001858.2002071

[7] Nikolaus Hansen. 2008. CMA-ES with Two-Point Step-Size Adaptation. *CoRR* abs/0805.0231 (2008). http://arxiv.org/abs/0805.0231

[8] Nikolaus Hansen. 2009. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers (GECCO '09)*. ACM, 2389–2396.

[9] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions*. Research Report RR-6829. INRIA. https://hal.inria.fr/inria-00362633

[10] Nikolaus Hansen and Andreas Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation (CEC), 1996 IEEE Congress on*. IEEE, 312–317.

[11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. *Sequential Model-Based Optimization for General Algorithm Configuration*. Springer Berlin Heidelberg, Berlin, Heidelberg, 507–523. DOI: http://dx.doi.org/10.1007/978-3-642-25566-3_40

[12] Grahame Jastrebski, Dirk V Arnold, and others. 2006. Improving evolution strategies through active covariance matrix adaptation. In *Evolutionary Computation (CEC), 2006 IEEE Congress on*. IEEE, 2814–2821.

[13] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory Landscape Analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11)*. ACM, New York, NY, USA, 829–836. DOI: http://dx.doi.org/10.1145/2001576.2001690

[14] A. Piad-Morffis, S. Estevez-Velarde, A. Bolufe-Rohler, J. Montgomery, and S. Chen. 2015. Evolution strategies with thresheld convergence. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*. 2097–2104. DOI: http://dx.doi.org/10.1109/CEC.2015.7257143

[15] John R. Rice. 1976. The Algorithm Selection Problem*. Advances in Computers, Vol. 15. Elsevier, 65 – 118. DOI: http://dx.doi.org/10.1016/S0065-2458(08)60520-3

[16] Sander van Rijn, Hao Wang, Matthijs van Leeuwen, and Thomas Bäck. 2016. Evolving the Structure of Evolution Strategies. *Computer* 49, 5 (May 2016), 54–63.

[17] Hao Wang, Michael Emmerich, and Thomas Bäck. 2014. Mirrored Orthogonal Sampling with Pairwise Selection in Evolution Strategies. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. ACM, 154–156.