Re-evaluating Algorithm Variations using Empirical Similarity

Jair Pereira Junior
pereira-junior.ua.ws@alumni.tsukuba.ac.jp
University of Tsukuba
Japan

Claus Aranha
caranha@cs.tsukuba.ac.jp
University of Tsukuba
Japan

ABSTRACT

In the context of metaheuristic search algorithms, two recent approaches have been proposed to measure algorithm similarity. The first one is based on shared search strategies, such as mutation or selection. The second one involves an empirical analysis that uses a set of performance metrics on benchmark problems. This paper explores whether high Component Similarity corresponds to Performance Similarity by analyzing these two similarity metrics on 9 CMA-ES variants on the COCO benchmark.

CCS CONCEPTS

• **Computing methodologies** → **Continuous space search**; • **Mathematics of computing** → **Evolutionary algorithms**.

KEYWORDS

metaheuristic, algorithm similarity, performance similarity

ACM Reference Format:
Jair Pereira Junior and Claus Aranha. 2023. Re-evaluating Algorithm Variations using Empirical Similarity. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590593

1 INTRODUCTION

Metaheuristic Search Algorithms are effective in solving black-box optimization problems. However, algorithms perform differently depending on differences in the landscape characteristics of these problems (separability, modality, conditioning, and others).

Many new metaheuristics and variants have been proposed aiming to exploit specific landscape characteristics. Nonetheless, the creation of numerous algorithms has been criticized for oversimplifying complex concepts [3, 14], making it hard to understand and compare algorithms. Therefore, a systematic evaluation and comparison of metaheuristics is required to determine their relative strengths and weaknesses, similarities and differences.

As this problem gained more attention, researchers introduced two new approaches to measure the similarity among metaheuristics. One approach is to decompose algorithms into search strategy components and measure the similarity based on the number of shared components [4].

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3590593

A different approach is the empirical similarity analysis. This approach uses the performance metrics of the metaheuristics on a set of benchmark problems as a measure of similarity [13].

In this work, we combine these two lines of algorithm analysis and investigate where they agree and differ. We apply these similarity metrics to 9 CMA-ES variants on the COCO Benchmark [1].

We found that combining these two similarity metrics is effective for analyzing the similarities and differences between metaheuristics, as it provides insights into the differences in Performance Profiles that can be attributed to specific components.

It is essential to emphasize that our objective is to analyze the component and Performance Similarity metrics rather than to determine the best algorithm.

The scripts to process the data and generate the figures are available online for reproducibility[1].

2 PRELIMINARIES

This section outlines the technical basis for our analysis.

2.1 Component Similarity

The Pool Template [4] is a framework to describe and decompose metaheuristics into components. Among its various structures to describe an algorithm, the Updating Mechanism stands out, as it specifies the process for updating the candidate solutions.

After decomposition, it is possible to measure the similarity of algorithm A to B as

$$CS(A,B) = \frac{|C_A \cap C_B|}{|C_A|} \tag{1}$$

where $|C_A \cap C_B|$ is the number of common components between A and B and $|CA|$ is the number of components in A. It is important to note that $CS(A,B)$ is not necessarily the same as $CS(B,A)$. In the original work, the CS is computed for each structure in the Pool Template and then summed with 55.5% weight for the Updating Mechanisms and 5.5% for all the other structures. After normalizing, the similarity ranges between 0 (high dissimilarity) and 1 (high similarity).

2.2 COCO Benchmark

The COCO Benchmark [10] offers a testbed with 24 single-objective noiseless functions in the continuous numerical domain*bbob suite*. These functions are grouped based on their main characteristics: (1) separability, (2) low or moderate conditioning, (3) high conditioning and unimodality, (4) multi-modality and adequate global structure, and (5) multi-modality and weak global structure [5].

[1] https://github.com/jair-pereira/mhsim_cmaes

In this benchmark, *problem instances* are transformations within the function space. These different instances provide a way to perform repetitions while preventing possible exploitation, like the optimum placement. When evaluating the performance of algorithms, the COCO platform records the number of function evaluations needed for the algorithm to reach each of the 51 targets for a given *problem instance*. The data from the 15 problem instances are aggregated for a given function in a given dimension to calculate the expected runtime (ERT). The ERT is a curve of the 51 targets x sum of all function evaluations divided by the number of instances where the target was reached divided by the dimension [9].

The COCO Benchmark provides an online experimental data archive of several algorithms. The following section presents the 9 algorithms chosen from this archive.

2.3 Metaheuristics

We chose as study case the variants of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Mainly, variants of the same algorithm share the same core but differ in only a few components, enabling us to investigate the impact of specific components on the overall performance of the algorithm. Likewise, CMA-ES has the advantage of not requiring parameter tuning due to its adaptive settings.

Several CMA-ES variants have been proposed to improve its performance or exploit specific problem characteristics. In this work, we analyze 9 CMA-ES variations recently evaluated on the COCO benchmark [1]. More details can be found in Section 3.1, where we decompose these methods and present their components in Table 1.

3 ALGORITHM SIMILARITY

Our goal is to assess if high CS (theoretical) suggest Performance Similarity (empirical). To this end, we first decomposed all the CMA-ES variants. Then, we used the decomposition to compute their Component Similarity. Lastly, we introduce the Performance Similarity and the quality indicator from the experimental data.

3.1 CMA-ES: Component Similarity

To compute the CS, we first decomposed the CMA-ES variants as shown in Table 1. For the decomposition, we considered only the characteristics that were clearly stated in their respective paper. After decomposition, the CS can be computed using Equation 1. As previously mentioned, the original Pool Template gives higher weight to the update mechanisms. Since we analyze only variations of the same algorithm, all components are given the same weight. The similarity values are shown in Figure 1.

3.2 Performance Similarity

The PS[13] is a way to compute the empirical similarity of two algorithms. The PS between algorithms A and B is defined as:

$$\underset{Similarity}{Performance}(A,B) = \frac{1}{1+\sqrt{\sum_{i=1}^{n}(q_i^A - q_i^B)^2}} \tag{2}$$

where n is the number of problems, and q is a quality indicator of the problem i. The similarity is between 0 and 1, where values near 0 mean high dissimilarity and values near 1 mean high similarity.

Figure 1: Component Similarity of the CMA-ES variants.

The original paper proposed the error to the optimum as a quality indicator. However, we use the expected runtime (ERT) because it integrates the error to the optimum and the effort to each it, allowing us to capture the entire convergence process instead of just the best function value found.

$$q_i^A = AREA(log_{10}(ERT)) \tag{3}$$

The empirical data used to compute the PS was downloaded from the COCO data archive [1] for problems with 10 dimensions.

4 ANALYSIS AND DISCUSSION

It is natural to think that if two algorithms use the exact same search strategies, they will perform exactly the same on all optimization problems. Considering this, we want to investigate if having similar search strategies yields a similar PS. In other others, we want to answer the question: Does high CS suggest PS?

To answer this question, we compare CS from Figure 1 to the PS in Figure 2. We analyze 6 scenarios, one considering all 24 functions and the other 5 considering each group of functions individually.Lastly, we use the Pearson correlation coefficient to assess the correlation between Component and PS. We added the RANDOMSEARCH in the figure to reference a completely different algorithm.

4.1 All 24 Functions and Group 5

This subsection analyzes the PS between the algorithms on all benchmark functions, as shown in Figure 2a.

First, PS is expected near 1 between CMA-ES-pycma and CMA-ES-Akimoto because they are the same algorithm despite different implementations. However, the score is only 0.37. This score is considered low because, in contrast, Akimoto's version is more similar to DD-CMA-ES-pycma (0.78) and DD-CMA-ES-Akimoto (0.64). Moreover, there is the same expectation of high PS between DD-CMA-ES-pycma and DD-CMA-ES-Akimoto. However, despite a higher score (0.65), it still does not reach a level of complete similarity, which would be represented by a score near 1. The low PS between CMA-ES-pycma and CMA-ES-Akimoto suggests an underlying difference in implementation since they use the same components. On the other hand, the higher PS in algorithms with

Table 1: Component decomposition of all CMA-ES variants

Algorithm	Recombination	CMA: Rank-One	CMA: Rank-μ	CMA: Active CM	CMA: Heavyside Function Stall	Restart Strategy	Diagonal	Keeping Search Points	Mirrored Sampling	Hessian Estimation
CMA-ES-Akimoto[2, 6]	Weighted	Yes	Yes	Yes	Yes	IPOP	fixed D=I	No	No	No
CMA-ES-pycma[6, 8]	Weighted	Yes	Yes	Yes	Yes	IPOP	fixed D=I	No	No	No
DD-CMA-ES-Akimoto[2, 6]	Weighted	Yes	Yes	Yes	Yes	IPOP	Adaptive	No	No	No
DD-CMA-ES-pycma[6, 8]	Weighted	Yes	Yes	Yes	Yes	IPOP	Adaptive	No	No	No
CMA-ES-APOP-Var1[11]	Weighted	No	No	Yes	No	APOP	fixed D=I	No	No	No
CMAES-APOP-KP[12]	Weighted	No	No	Yes	No	APOP	fixed D=I	Yes	No	No
CMAES-APOP-MA[12]	Weighted	No	No	Yes	No	APOP	fixed D=I	No	Yes	No
CMAES-APOP-KMA[12]	Weighted	No	No	Yes	No	APOP	fixed D=I	Yes	Yes	No
HE-ES[7]	Weighted	No	No	Yes	No	IPOP	fixed D=I	No	Yes	Yes

the Adaptive Diagonal Decoding component suggests that this component smoothens out implementation differences.

Second, the group CMA-ES-Akimoto, DD-CMA-ES-Akimoto, and DD-CMA-ES-pycma are more empirically similar to CMAES-APOP-KP (0.55, 0.61, and 0.57, respectively) than to CMA-ES-pycma (0.37, 034, and 036, respectively). It could be expected that the former group is more similar to CMA-ES-pycma since they have much more common components (CS of 1.00, 0.86, and 0.86, respectively) than to CMAES-APOP-KP (CS of 0.43, 0.29, and 0.29, respectively).

Third, Nguyen's implementations (var1, KP, MA, KMA) have a high CS as shown in Figure 1, but very low PS. These variants differs on one or two components, meaning that these differences drastically affect their Performance Profiles.

Functions in Group 5 (Figure 2d) seem to have greater weight when computing the PS on all 24 functions, as they have a similar pattern and scores. Thus the observations above are the same.

4.2 Functions in Group 1

Figure 2b shows the PS for Separable Functions, which has a similar pattern as in 2a, but with higher similarity values. An exception is CMAES-APOP-Var1, now having high similarity performance with CMAES-APOP-KP and the group CMA-ES-pycma, CMA-ES-Akimoto, DD-CMA-ES-pycma, and DD-CMA-ES-Akimoto.

This time, CMA-ES-pycma and CMA-ES-Akimoto have the expected high similarity (0.89), and DD-CMA-ES-pycma and DD-CMA-ES-Akimoto have a relatively high score (0.67). Meaning that their implementation differences are less affected on Group 1.

Second, Mirroed Sampling seem to affect the Performance Profiles to a high extent on *Separable functions*. This is demonstrated by the high correlation score of 0.92 between CMAES-APOP-MA and CMAES-APOP-KMA, as well as the fact that CMA-ES-KP and CMAES-APOP-var1 both exhibit a high score of 0.92, but have considerably lower scores (0.34) compared to CMAES-APOP-MA and CMAES-APOP-KMA. Notably, the main distinguishing feature between the algorithms in the latter group and the former group is the Mirrored Sampling component. It is important to comment that HE-ES also implements Mirrored Sampling but has a low similarity score to Nguyen's implementations.

4.3 Functions in Groups 2, 3, and 4

Figure 2c shows the PS for Functions with low or moderate conditioning. Groups 3 and 4 share a clear pattern with Group 2 and are omitted from the figures in this section due to limited space.

Table 2: Pearson Correlation Index between the Theoretical Similarity and Empirical Similarity for the CMA-ES variants.

Functions	Pearson Correlation	P-value
All	0.50	1.68e-6
Group 1	0.49	3.73e-6
Group 2	0.55	9.43e-8
Group 3	0.30	7.36e-3
Group 4	0.33	2.63e-3
Group 5	0.48	4.76e-6

In these 3 groups, the studied algorithms have a very high similarity index (over 0.8). Due to these uniformly high scores among all the algorithms, it is challenging to draw meaningful observations.

4.4 Similarity Correlation

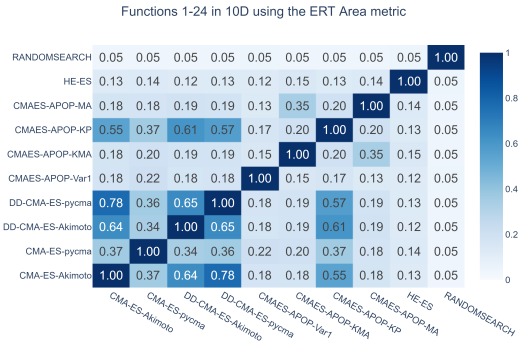
In this subsection, we analyze the correlation between the CS, presented in Figure 1, and the PS by groups of functions, presented in Figure 2a. To this end, we use the Pearson Correlation index as shown in Table 2. As can be seen, the correlation index indicates a moderate positive (0.5>x) relationship between the theoretical and empirical measures for Groups 1, 2, and 5; and a weak positive relationship for Groups 3 and 4 (0.3<x<0.5). This interesting result suggests that CS can provide some insights into PS. However, the weaker correlation observed in Groups 3 and 4 may indicate other factors affecting the Performance Profiles.

4.5 Discussion

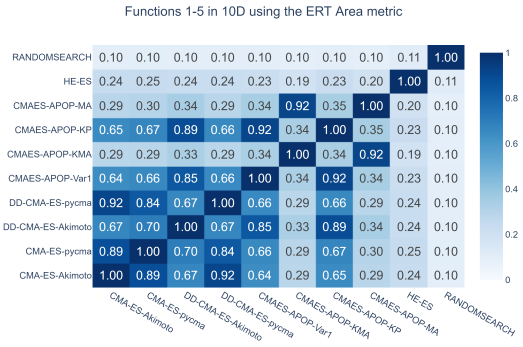
We had some general observations. First, it is important to compute the PS per group of benchmark functions. This is because we found that Group 5 had a greater impact on the overall similarity measure, as shown in Figure 2a and Figure 2d, while other groups showed a different pattern. Second, we observed instances where CS did not align with PS, such as when different people implemented the exact same algorithm resulting in unexpectedly low PS scores. Another example was the effect of Mirrored Sampling on *Separable functions*. Lastly, our CS and PS analysis via the Pearson Correlation Index has shown a weak to moderate correlation. Overall, evaluating metaheuristics using theoretical and empirical measures is important, as the former may not always reflect performance.

5 CONCLUSION

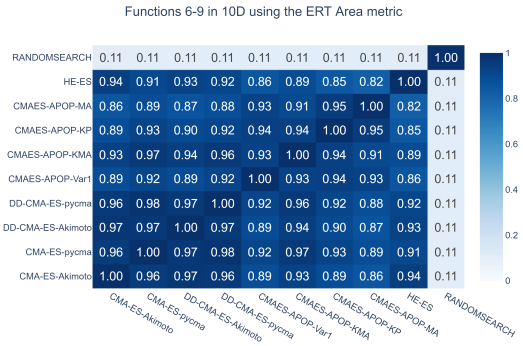
The goal of our study was to analyze the relationship between CS and PS. Our findings suggest that combining CS and PS metrics is effective for comparing metaheuristics. This enables us to gain



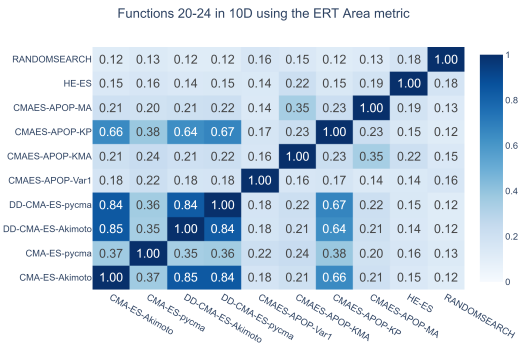
(a) All 24 benchmark functions



(b) Separable functions



(c) Functions with low or moderate conditioning



(d) Multi-modal functions with weak global structure

Figure 2: Performance Similarity of the CMA-ES variants divided by classes of problems.

insights into the differences in Performance Profiles that can be attributed to specific components. Notably, we found that these metrics exhibit a weak to moderate correlation, as determined by Pearson's correlation coefficient.

However, the algorithm decomposition process is crucial in determining the comparison results. Our study showed instances where the expected PS based on CS was not achieved. Overall, this highlights the importance of the algorithm description and decomposition. A standardized method of decomposing metaheuristics into components would facilitate the metaheuristics comparison and the use of CS. Additionally, when publishing a new metaheuristic or a variant, it would be insightful for the authors to offer its decomposition and relevant implementation details, as they can greatly impact the algorithm's performance.

This study has limitations, such as considering only functions in 10D. Further research could extend the analysis to other dimensions, as the dimensionality can significantly impact a problem's difficulty. Additionally, it is worth noting that the relationship between CS and PS may not be linear, as components can have both constructive and destructive effects on performance. Future work could explore these factors and their impact on the two similarity metrics.

REFERENCES

[1] [n. d.]. BBOB Data Archive. https://numbbo.github.io/data-archive/bbob/

[2] Youhei Akimoto and Nikolaus Hansen. 2020. Diagonal acceleration for covariance matrix adaptation evolution strategies. *Evolutionary computation* 28, 3 (2020), 405–435.

[3] Claus Aranha, Christian L Camacho Villalón, Felipe Campelo, Marco Dorigo, Rubén Ruiz, Marc Sevaux, Kenneth Sörensen, and Thomas Stützle. 2022. Metaphor-based metaheuristics, a call for action: the elephant in the room. *Swarm Intelligence* 16, 1 (2022), 1–6.

[4] Jesica de Armas, Eduardo Lalla-Ruiz, Surafel Luleseged Tilahun, and Stefan Voß. 2022. Similarity in metaheuristics: a gentle step towards a comparison methodology. *Natural Computing* 21, 2 (2022), 265–287.

[5] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. 2010. *Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions.* Technical Report. Citeseer.

[6] Mohamed Gharafi. 2022. Benchmarking of two implementations of CMA-ES with diagonal decoding on the bbob test suite. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion.* 1700–1707.

[7] Tobias Glasmachers and Oswin Krause. 2020. The Hessian estimation evolution strategy. In *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I 16.* Springer, 597–609.

[8] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. 2019. CMA-ES/pycma on Github. Zenodo, DOI: 10.5281/zenodo. 2559634.(Feb. 2019).

[9] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. 2021. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software* 36, 1 (2021), 114–144.

[10] Nikolaus Hansen, Dimo Brockhoff, Olaf Mersmann, Tea Tusar, Dejan Tusar, Ouassim Ait ElHara, Phillipe R. Sampaio, Asma Atamna, Konstantinos Varelas, Umut Batu, Duc Manh Nguyen, Filip Matzner, and Anne Auger. 2019. *COmparing Continuous Optimizers: numbbo/COCO on Github.* https://doi.org/10.5281/zenodo.2594848

[11] Duc Manh Nguyen. 2018. Benchmarking a variant of the CMAES-APOP on the BBOB noiseless testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion.* 1521–1528.

[12] Duc Manh Nguyen. 2022. Benchmarking some variants of the CMAES-APOP using keeping search points and mirrored sampling combined with active CMA on the BBOB noiseless testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion.* 1734–1742.

[13] Jair Pereira Junior and Claus Aranha. 2022. Empirical Similarity Measure for Metaheuristics. In *Bioinspired Optimization Methods and Their Applications: 10th International Conference, BIOMA 2022, Maribor, Slovenia, November 17–18, 2022, Proceedings.* Springer, 69–83.

[14] Kenneth Sörensen. 2015. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research* 22, 1 (2015), 3–18.