ClustOpt: A Clustering-based Approach for Representing and Visualizing the Search Dynamics of Numerical Metaheuristic Optimization Algorithms

1st Gjorgjina Cenikj
Computer Systems Department
Jožef Stefan Institute
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia
gjorgjina.cenikj@ijs.si

2nd Gašper Petelin
Computer Systems Department
Jožef Stefan Institute
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia
gasper.petelin@ijs.si

3rd Tome Eftimov
Computer Systems Department
Jožef Stefan Institute
Ljubljana, Slovenia
tome.eftimov@ijs.si

***Abstract*—Understanding the behavior of numerical metaheuristic optimization algorithms is critical for advancing their development and application. Traditional visualization techniques, such as convergence plots, trajectory mapping, and fitness landscape analysis, often fall short in illustrating the structural dynamics of the search process, especially in high-dimensional or complex solution spaces. To address this, we propose a novel representation and visualization methodology that clusters solution candidates explored by the algorithm and tracks the evolution of cluster memberships across iterations, offering a dynamic and interpretable view of the search process. Additionally, we introduce two metrics – algorithm stability and algorithm similarity – to quantify the consistency of search trajectories across runs of an individual algorithm and the similarity between different algorithms, respectively. We apply this methodology on a set of ten numerical metaheuristic algorithms, revealing insights into their stability and comparative behaviors, thereby providing a deeper understanding of their search dynamics.**

***Index Terms*—algorithm trajectory representations, optimization algorithm analysis**

1. Introduction

Visualization techniques are a critical means of shedding light on the behavior of metaheuristic numerical optimization algorithms. Conventional methods such as convergence analysis, trajectory visualizations, and fitness landscape analysis provide valuable insights into aspects like convergence speed, diversity, and solution quality. However, these approaches often fail to capture the structural dynamics of the search process, particularly in high-dimensional or complex spaces. Existing methods rarely address the location of the solution candidates in the search space, which can reveal crucial information about the exploratory and exploitative strategies of an algorithm.

We propose ClustOpt, a novel representation and visualization methodology for metaheuristic numerical population-based optimization algorithms, that focuses on clustering solution candidates explored by optimization algorithms. The methodology groups these candidates based on their similarity in the solution space and visualizes the evolution of cluster memberships across iterations. It provides a dynamic and interpretable representation of the search process, highlighting patterns such as the emergence, dominance, or disappearance of clusters over time. The resulting visualizations offer insights into the algorithms' navigation of the search space, enabling a deeper understanding of their behavior. Unlike traditional visualization techniques that focus solely on visualizations, our approach transforms visualizations into structured representations that allow for efficient comparison across multiple algorithms. It enables large-scale benchmarking of optimization algorithms' sensitivity to initialization and the analysis of their similarity based on detailed search trajectories. We demonstrate the methodology's applicability for analyzing the behaviour of 10 single-objective continuous optimization algorithms on the Black Box Optimization Benchmarking (BBOB) [1] suite.

The remainder of the paper is organized as follows. In Section 2, we present the related work, while in Section 3 we explain the methodology of constructing the representations. In Section 4 we demonstrate how the representations can be visualized and used for different tasks. Finally, Section 5 concludes the paper and specifies directions for future work.

Reproducibility: The code is available at https://github.com/gjorgjinac/ClustOpt.

2. Related work

A prevalent method for analyzing the behaviour of single-objective continuous optimization algorithms is convergence analysis [2], [3], which seeks to determine

whether an algorithm converges to the global optimum and the speed at which it achieves convergence. However, this approach focuses on the quality of the solutions, ignoring their location in the search space.

One of the biggest challenges of visualizing the solutions explored by an algorithm during the search procedure is the large number of solutions being explored, especially when the problem is high dimensional. Consequently, several works have already attempted to reduce the high dimensional data into two dimensions using different dimensionality reduction techniques such as Principal Component Analysis [4], [5], t-Distributed Stochastic Neighbor Embedding (t-SNE) [4], [6], Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [4], Sammon mappings [7], [8], and Self-Organizing Maps [9]. An analysis of the ability of different dimensionality reduction techniques to preserve meaningful information about the population dynamics can be found in [4]. While such approaches are useful for visualization purposes, they do not allow an automatic comparison of algorithm behavior and may be too overwhelming for manual examination when numerous algorithms, problems, and executions are compared.

Apart from methodologies based on dimensionality reduction, graph visualization approaches have also been proposed. For example, Local Optima Networks (LONs) [10] represent fitness landscapes in terms of a graph where local optima are nodes and search transitions are edges defined by an exploration search operator. Similarly, Search Trajectory Networks (STNs) [11], are graph-based visualizations where nodes represent locations of the search space, not limited to local optima, while the edges signify the progression between these locations. In this case, the search space is partitioned into a predefined set of locations and an algorithm's optimization process is visualized by taking the locations of representative solutions from each iteration and connecting them into a graph structure. While our approach bares similarities to the STNs, the main difference is that we target the representation of the entire search trajectory (all candidate solutions within each population), while the STNs are representing only a single representative solution from each population (usually the best solution from each population). Additionally, STNs produce a graph-based visualization, while our approach produces a numerical representation of the trajectory which allows it to be visualized and compared to other trajectories using machine learning (ML) approaches.

3. Methodology

We propose a methodology for visualizing algorithm trajectories and representing them in terms of numerical features. Given a fixed optimization problem p of dimension d, and a set population-based algorithms $A = \{a_1, a_2, \dots\}$, executed multiple times with random seeds $R = \{r_1, r_2, \dots\}$ for a budget of b iterations with a population size s on the same problem, our proposed approach enables the analysis and comparison of the algorithm search behaviour. The proposed approach can be used to analyze a single algorithm trajectory($|A| = 1$, $|R| = 1$), or multiple algorithm trajectories - different executions of the same algorithm ($|A| = 1$, $|R| > 1$) or executions of different algorithms ($|A| > 1$, $|R| \geq 1$).

It consists of four main steps:

- Merging trajectories - The candidate solutions explored by all algorithms in all iterations of all executions are merged into a set S containing $|A| \times |R| \times b \times s$ solution vectors of dimension d.
- Scaling - To account for the fact that the d different dimensions of the $\mathbf{x} \in S$ candidate solutions may have different ranges of values, we first scale all candidate solutions to be in the range [0,1]. The scaling is done on the merged set of trajectories jointly (i.e., including the trajectories for all algorithms on all problem with different seeds) in order to allow a comparison of the results for different trajectories.
- Clustering - The scaled candidate solutions are then clustered into c clusters. The search space is thereby partitioned into areas of interest which are shared across all executions of the algorithms. In our experiments, we determine the number of clusters using the elbow method and apply k-means clustering with Euclidean distance. We selected k-means clustering because it produces more spherical clusters, which are generally easier to interpret and analyze. We use a k-means++ initialization, which has been shown to speed up convergence and often improve the quality of clusters.
- Representation calculation - The proposed representation is based on the number of solutions from each iteration of the trajectory which belong to each cluster. For this purpose, for each algorithm, execution, and iteration of the search process, we count the number of candidate solutions from the population generated by the algorithm in the particular iteration which belong to each of the c clusters. The representation of a single trajectory is therefore a vector of $b \times c$ values, indicating the number of candidate solutions from each of the b iterations which are placed into each of the c clusters.

4. Results

We demonstrate the applicability of the proposed methodology for the following purposes: *i)* Visualizing algorithm trajectories; *ii)* Analyzing the sensitivity of the optimization algorithm to its initialization; *iii)* Analyzing the similarity of different algorithms.

4.1. Data

We demonstrate the proposed methodology by analyzing the trajectories of 10 algorithms from the MEALPY [12] Python library on the Black Box Optimization Benchmarking (BBOB) benchmark [1]. We use the first five instances from each of the 24 BBOB problems in dimensions $d = \{2, 5, 10\}$. The instances from the same problem class differ in shifting, scaling, and/or rotation. The algorithms are executed with their default parameters, a population size of 50 and a budget of $10d$ iterations ($500d$ function evaluations).

In particular, we select the following algorithms for analysis: BaseDE (Differential Evolution) [13], SADE (Self-Adaptive Differential Evolution) [14], JADE (Adaptive Differential Evolution With Optional External Archive) [15], SHADE (Success-history based parameter adaptation for Differential Evolution) [16], EnhancedAEO (Enhanced Artificial Ecosystem-based Optimization) [17], ModifiedAEO (Modified Artificial Ecosystem-based Optimization) [18], OriginalAEO (Artificial Ecosystem-based Optimization) [19], AugmentedAEO (Augmented Ecosystem-based Optimization) [20], HI_WOA (Hybrid Improved Whale Optimization Algorithm) [21], and OriginalWOA (Whale Optimization Algorithm) [22]. We select these particular algorithms because they are variants of the Differential Evolution, Artificial Ecosystem-based Optimization, and the Whale Optimization Algorithm. We therefore expect some of the variants of the same algorithm to have similar behaviour, and we use this to confirm the validity of our representations. However, our representations are applicable to any population-based continuous optimization algorithm.

4.2. Visualizing algorithm trajectories

In this section, we demonstrate how the proposed methodology can be used for visualizing a single algorithm trajectory and comparing it to trajectories of other algorithms. We start off with an example of how the proposed representations can be visualized. We demonstrate the example on $2d$ problems, since it is straightforward to visualize such problems without a loss of information. However, the methodology is applicable to problems of higher dimensions, and in the following subsections we include analysis on $5d$ and $10d$ problems. Figure 1 depicts the representation of the OriginalAEO, AugmentedAEO, and SHADE algorithms on the first instance of the 5th $2d$ BBOB problem (Linear Slope), while Figure 2 shows their behaviour on the first instance of the 16th $2d$ BBOB problem (Weierstrass).

Each subplot refers to a different algorithm. The horizontal axis shows the cluster number, as well as the location of the cluster centroid. The vertical axis depicts the iteration number. The values in the heatmap cells denote the number of candidate solutions which were

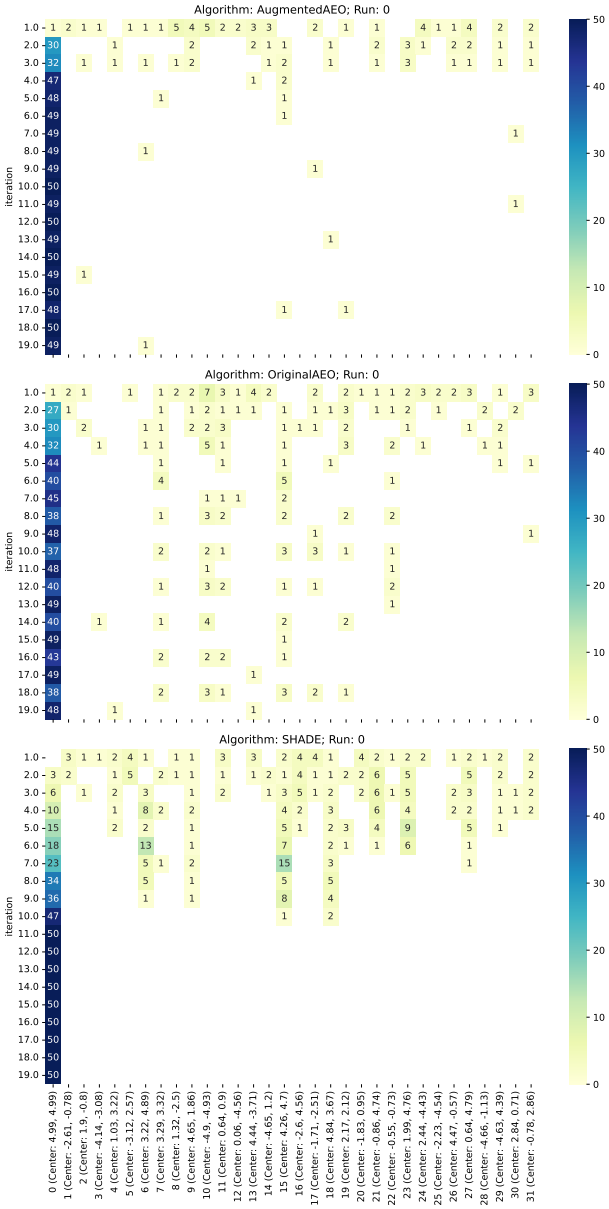


Figure 1: Visualizations of the trajectories of the OriginalAEO, AugmentedAEO and SHADE algorithms on the first instance of the fifth $2d$ BBOB problem (Linear Slope)

located in a particular cluster in a particular iteration. White cells denote empty clusters. The maximal value in the heatmap can be 50, i.e., the population size.

Comparing the visualizations of the three algorithms on the 5th problem in Figure 1, we can see that in the first iterations, they are all exploring different clusters, however, they all converge to cluster zero with centroid (4.99,4.99). This is expected since the problem visualized is the linear slope function, which is convex and very simple to solve. On the other hand, looking at the algorithms' behaviour on the Weierstrass problem

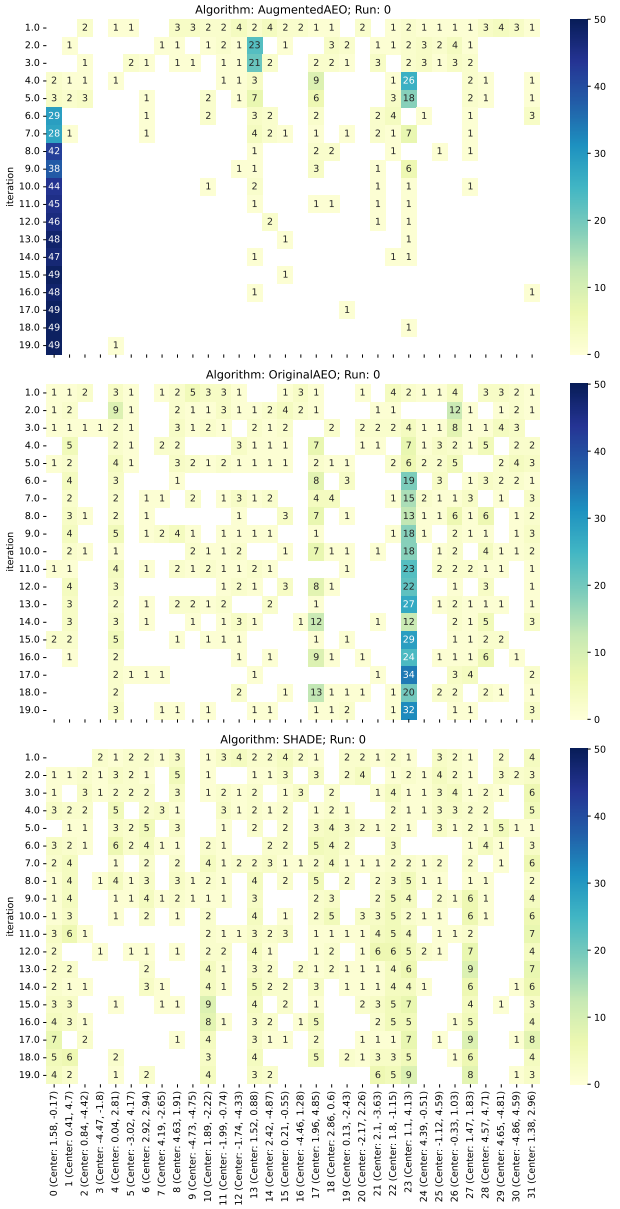


Figure 2: Visualizations of the trajectories of the OriginalAEO, AugmentedAEO and SHADE algorithms on the first instance of the 16th $2d$ BBOB problem (Weierstrass)

in Figure 2, we can see that the three algorithms have completely different outcomes. In the last iteration, the OriginalAEO algorithm has the majority of the candidate solutions in cluster 23, but is still doing some exploration of the other clusters. The AugmentedAEO algorithm ends up converging in cluster zero, with only a single candidate solution in cluster three. Finally, the SHADE algorithm is exploring multiple clusters and not focusing on a particular one. This behaviour is due to the fact that the Weierstrass function is highly rugged and moderately repetitive, and its global optimum is not

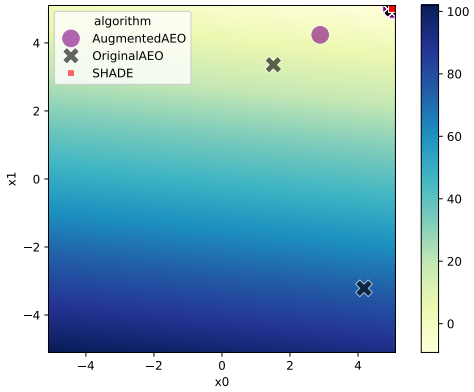


Figure 3: Visualization of the fifth $2d$ BBOB problem (Linear Slope) and the candidate solutions explored by the OriginalAEO, AugmentedAEO and SHADE algorithms in the last iteration

unique [1].

To validate our representations, we visualize the solutions explored by the three algorithms in two dimensions. Figures 3 and 4 contain a visualization of the two previously investigated functions (Linear Slope and Weierstrass). The horizontal and vertical axes denote the two dimensions of the candidate solution, while the color reflects the objective function value of the solution (lighter is lower and better). The location of the candidate solutions explored by the three algorithms in the last iteration of the search is denoted in the form of points on top of the color map. In Figure 3, most of the solutions are located in the upper right corner and are overlapping, meaning that the algorithms converge to the same solution, which we also saw in Figure 1. The AugmentedAEO algorithm has one candidate solution which is different from the majority of overlapping solutions located in the upper right corner, while the OriginalAEO algorithm has two such candidate solutions. This is also consistent with our findings in Figure 1, where these two algorithms had exactly the same number of candidate solutions located in different clusters.

Focusing on Figure 4, containing the same visualization for the Weierstrass problem, we can see that in this case, most of the candidate solutions are not overlapping. The exception is the AugmentedAEO algorithm, which has most of the candidate solutions overlapping around the point (1.8, 0) and a single differing solution around the point (0.5, 3.8). The other two algorithms have solutions in different area of the search space. This is also consistent with Figure 2.

4.3. Analyzing the sensitivity of the optimization algorithm to its initialization

In this subsection, we demonstrate the use of the proposed representations to evaluate the sensitivity of

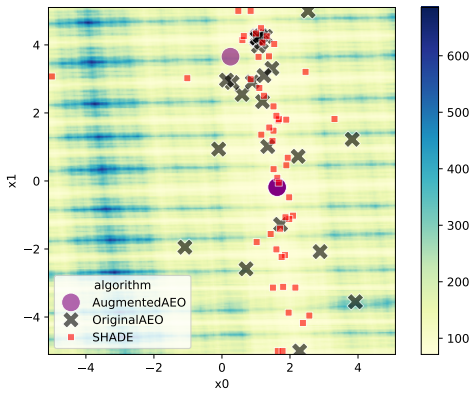


Figure 4: Visualization of the 16th $2d$ BBOB problem (Weierstrass) and the candidate solutions explored by the OriginalAEO, AugmentedAEO and SHADE algorithms in the last iteration

the optimization algorithm to its initialization, which we further refer to as stability. To this end, we analyze the execution of a single algorithm across different runs (multiple executions with a different initial population). We do this by aggregating the cosine similarity of the proposed trajectory representation for trajectories obtained by a single algorithm executed on a single problem with different random seeds which impact the initialization of the initial population, or more precisely:

$$stability(a,p) = \frac{1}{|R|(|R|-1)} \sum_{\substack{r_k,r_l \in R \\ r_k \neq r_l}} sim(v_{a,p,r_k}, v_{a,p,r_l}) \quad (1)$$

where a denotes an algorithm, p denotes a problem, R is the set of all random seeds used for different algorithm executions, sim represents the cosine similarity metric, while v_{a,p,r_k} is the vector representation of the trajectory of algorithm a executed on problem p using the random seed r_k. The intuition behind this metric is that algorithms which have high similarity of trajectory representations when executed multiple times on the same problem are more stable than ones with lower similarities.

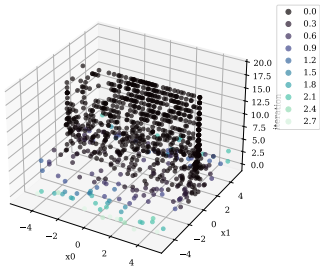
As an example, we demonstrate the behaviour of the AugmentedAEO and BaseDE algorithms on the first instance of the second BBOB $2d$ problem class. According to our analysis, AugmentedAEO has a relatively unstable behaviour on this problem instance, with a stability score of 0.32. Conversely, the BaseDE algorithm has a stable behaviour with a stability score of 0.81.

Figure 5 shows the trajectories of these two algorithms produced in two different executions on the first instance of the second BBOB $2d$ problem class. The visualization is generated following [4], with the exception that we do not perform any dimensionality reduction, since we have a $2d$ problem. Each subplot contains a single trajectory. The axes on the bottom, with range [-5,5], represent the values of the candidate solutions in the two dimensions. The vertical axis, with range [0,20], represents the iteration number, while the color indicates the objective function value of the candidate solution. Comparing figures 5a and 5b, plotting two runs of the BaseDE algorithm obtained with two different random seeds, we can see that the distribution of the candidate solutions is similar. In both runs, the algorithm does not converge to a single solution, but finds several solutions of similar quality. The locations of the explored candidate solutions are similar in both runs, regardless of the starting population. On the contrary, comparing the two executions of the AugmentedAEO algorithm in figures 5c and 5d, we can see a differing search pattern. In Figure 5c, after the fifth iteration, the AugmentedAEO algorithm is mainly doing exploitation around the point (1.2, 0.4). On the other hand, in Figure 5d, from iteration 3 to iteration 11, most of the candidate solutions are located around the point (-3.2, 0.4). In iterations 12 to 16, the algorithm jumps to another area of the search space, located around the point (1.3, 0.4). This means that the behaviour of the AugmentedAEO algorithm is more affected by the initialization of the first population, i.e., it is less stable than the BaseDE algorithm.

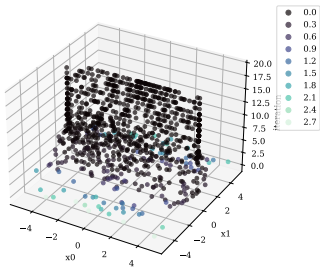
Figures 6 and 7 show the mean algorithm stability scores (aggregated across all instances of a problem class) for each algorithm on each of the BBOB problems in dimension 2, and 5, respectively. We can see that there are problems where all of the algorithms have a relatively stable behaviour, and problems where some algorithms experience lower stability. From Figure 6 we can see that in dimension 2, most algorithms are relatively stable on problems 1, 5, 6, 14, 17 and 21. This is especially pronounced for problems 1 and 5, where all algorithms have stability scores above 0.85. Considering the simplistic nature of these two functions, it does make sense that most algorithms achieve stable behavior and smoothly find the optimum across all runs. From Figure 7 we observe that in dimension 5, the only two problems where all algorithms are relatively stable are 1 and 5. In this dimension the DE-based algorithms (BaseDE, JADE, SADE and SHADE) have much higher stability scores than the other algorithms, which are more affected by the initialization. The same pattern is also present for $10d$ problems, although we omit the figure due to space limitations. This also makes sense, since the problem dimension increases the complexity and size of the search space, likely resulting in a larger number of clusters.

4.4. Analyzing the similarity of different optimization algorithms

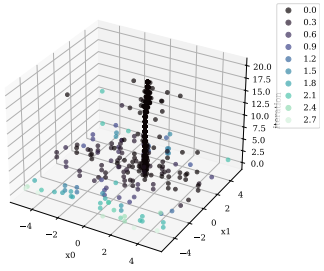
When calculating the similarity of different optimization algorithms, we take the mean similarity of



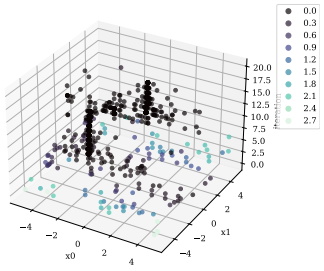
(a) BaseDE algorithm, first run



(b) BaseDE algorithm, second run



(c) AugmentedAEO algorithm, first run



(d) AugmentedAEO algorithm, second run

Figure 5: Trajectories of the BaseDE and AugmentedAEO algorithms executed three times on the first instance of the second BBOB $2d$ problem class

the algorithm trajectories produced by a pair algorithms on the same problem using the same random seed (starting with the same initial population). In particular, the similarity of a pair of algorithms a_m and a_n is calculated as:

$$similarity(a_m, a_n) = \frac{1}{|R||P|} \sum_{r \in R} \sum_{p \in P} sim\left(v_{a_m,p,r}, v_{a_n,p,r}\right) \quad (2)$$

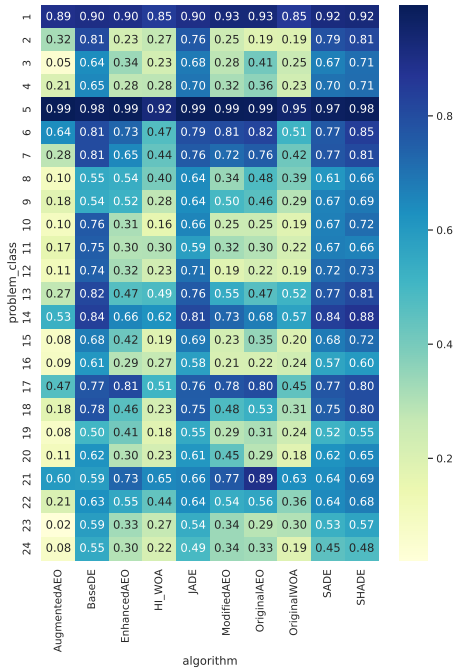


Figure 6: Algorithm stability per problem, $2d$

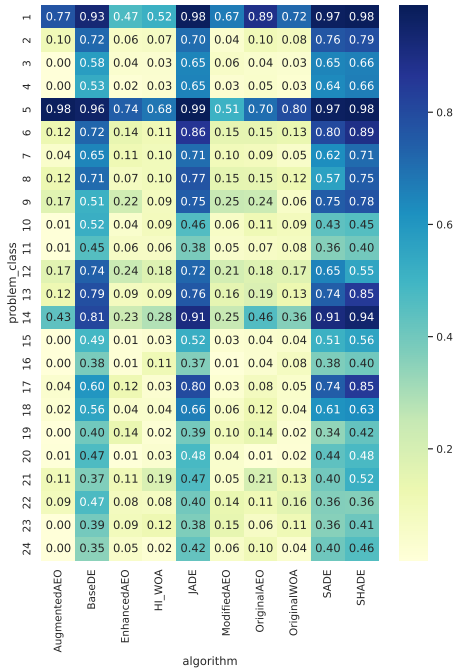


Figure 7: Stability of the investigated algorithms on the $5d$ BBOB problems, averaged across problem instances

where R is the set of all random seeds used for algorithm execution, P is the set of all problems, sim is the cosine similarity metric, while $v_{a_m,p,r}$ is the vector representation of the trajectory of algorithm a_m executed on problem p using the random seed r. Note that here we are comparing trajectories of different algorithms initialized with the same population (same random seed r), whereas in Equation 1 we are comparing trajectories of the same algorithm initialized with different populations (different random seeds). We do this since when we are measuring stability, we are interested in the behaviour of a single algorithm, across different initializations. On the other hand, when measuring the similarity of two algorithms, we want to eliminate the effect of the random initialization on the produced trajectories, so we therefore compare the algorithms starting from the same initial population, i.e, same random seed.

Figure 8 presents the mean algorithm similarity on $2d$, and $10d$ problems. In the first subplot, containing the similarity on $2d$ problems, we can see that the DE-based algorithms (BaseDE, SADE, JADE and SHADE) are clustered together and have similarity scores in the range 0.64 – 0.72. The HI_WOA and OriginalWOA algorithms also have a quite high similarity score of 0.66. Three of the AEO-based algorithms (EnhancedAEO, ModifiedAEO and OriginalAEO) are clustered together and have a similarity score of 0.58. Interestingly, the AugmentedAEO algorithm bears a somewhat lower similarity to the other AEO variants. We can also observe that the similarities of variants of the same algorithm are generally higher than the similarities of variants of a different algorithm, which indicates that our proposed representations are able to capture algorithm similarities in a meaningful way. Focusing on the results for $10d$ problems, we can see that the DE variants and the WOA variants are still clustered together, although with somewhat lower similarity scores. On the other hand, the AEO variants have relatively low similarity scores with any other algorithms. This is to some degree consistent with our analysis of their stability, where these algorithms become unstable in higher problem dimensions. An additional analysis of the execution time of each step of the methodology can be found in our github repository.

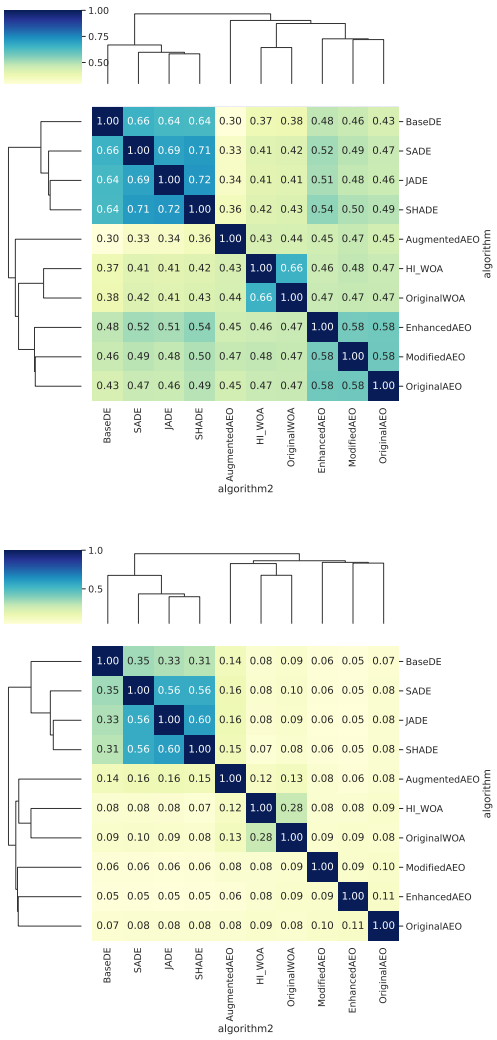


Figure 8: Algorithm similarity across problem dimensions for $2d$ (top), and $10d$ (bottom) problems.

5. Conclusion

This paper introduced ClustOpt, a novel vectorized representation and visualization methodology for analyzing the search process of optimization algorithms by clustering solution candidates and tracking their cluster transitions across iterations. This approach facilitated the definition of two key metrics: stability, which quantifies the consistency of an algorithm's trajectories across different initial populations, and similarity, which measures the resemblance of search trajectories between different algorithms. By offering a means to track, quantify, and visualize the dynamic behaviors of optimization algorithms, our framework provides valuable insights into their exploratory and exploitative characteristics, sensitivity to initialization, and comparative behaviors, thereby enabling a deeper understanding and refinement of optimization strategies. For future work, we are planning to perform a comprehensive analysis of the entire MEALPY Python library, to empirically find variants of the same algorithm leading to the same outcomes, so it can be a step further in addressing the call for action related to the methaphor-based metaheuristics. Finally, we would like to point out that this is an introductory study to the proposed methodology and a more detailed analysis of the impact of the clustering algorithm, its parameters, and similarity metric would be useful to further understand its behaviour, outcomes and limitations.

Acknowledgments

We acknowledge the support of the Slovenian Research and Innovation Agency through program grant No.P2-0098, young researcher grants No.PR-12393 to GC and No.PR-11263 to GP, and project grants No.J2-4460 and No.GC-0001. This work is also funded by the European Union under Grant Agreement No.101187010 (HE ERA Chair AutoLearn-SI) and the EU Horizon Europe program (grant No.101077049, CONDUCTOR).

References

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

[2] T. Cai and H. Wang, "A general convergence analysis method for evolutionary multi-objective optimization algorithm," *Information Sciences*, vol. 663, p. 120267, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025524001804

[3] P. Chakraborty, S. Sharma, and A. K. Saha, "Convergence analysis of butterfly optimization algorithm," *Soft Computing*, vol. 27, no. 11, p. 7245–7257, Apr. 2023. [Online]. Available: http://dx.doi.org/10.1007/s00500-023-07920-8

[4] A. De Lorenzo, E. Medvet, T. Tušar, and A. Bartoli, "An analysis of dimensionality reduction techniques for visualizing evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1864–1872. [Online]. Available: https://doi.org/10.1145/3319619.3326868

[5] T. D. Collins, "Applying software visualization technology to support the use of evolutionary algorithms," *Journal of Visual Languages and Computing*, vol. 14, no. 2, pp. 123–150, 2003.

[6] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[7] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. 100, no. 5, pp. 401–409, 1969.

[8] H. Pohlheim, "Multidimensional scaling for evolutionary algorithms - visualization of the path through search space and solution space using sammon mapping," *Artificial Life*, vol. 12, no. 2, pp. 203–209, 2006.

[9] H. B. Amor and A. Rettinger, "Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation," in *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '05*. ACM, 2005, pp. 1531–1538.

[10] G. Ochoa, S. Verel, F. Daolio, and M. Tomassini, "Local optima networks: A new model of combinatorial fitness landscapes," *Recent advances in the theory and application of fitness landscapes*, pp. 233–262, 2014.

[11] G. Ochoa, K. M. Malan, and C. Blum, "Search trajectory networks of population-based algorithms in continuous spaces," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2020, pp. 70–85.

[12] N. Van Thieu and S. Mirjalili, "Mealpy: An open-source library for latest meta-heuristic algorithms in python," *Journal of Systems Architecture*, vol. 139, p. 102871, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762123000504

[13] R. Mallipeddi, P. Suganthan, Q. Pan, and M. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, p. 1679–1696, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2010.04.024

[14] A. Qin and P. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *2005 IEEE Congress on Evolutionary Computation*, vol. 2. IEEE, p. 1785–1791. [Online]. Available: http://dx.doi.org/10.1109/CEC.2005.1554904

[15] J. Zhang and A. Sanderson, "Jade: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, p. 945–958, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2009.2014613

[16] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, Jun. 2013, p. 71–78. [Online]. Available: http://dx.doi.org/10.1109/CEC.2013.6557555

[17] A. Eid, S. Kamel, A. Korashy, and T. Khurshaid, "An enhanced artificial ecosystem-based optimization for optimal allocation of multiple distributed generations," *IEEE Access*, vol. 8, p. 178493–178513, 2020. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2020.3027654

[18] A. S. Menesy, H. M. Sultan, A. Korashy, F. A. Banakhr, M. G. Ashmawy, and S. Kamel, "Effective parameter extraction of different polymer electrolyte membrane fuel cell stack models using a modified artificial ecosystem optimization algorithm," *IEEE Access*, vol. 8, p. 31892–31909, 2020. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2020.2973351

[19] W. Zhao, L. Wang, and Z. Zhang, "Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm," *Neural Computing and Applications*, vol. 32, no. 13, p. 9383–9425, Sep. 2019. [Online]. Available: http://dx.doi.org/10.1007/s00521-019-04452-x

[20] N. Van Thieu, S. Deb Barma, T. Van Lam, O. Kisi, and A. Mahesha, "Groundwater level modeling using augmented artificial ecosystem optimization," *Journal of Hydrology*, vol. 617, p. 129034, Feb. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.jhydrol.2022.129034

[21] C. Tang, W. Sun, W. Wu, and M. Xue, "A hybrid improved whale optimization algorithm," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*. IEEE, Jul. 2019. [Online]. Available: http://dx.doi.org/10.1109/ICCA.2019.8900003

[22] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, p. 51–67, May 2016. [Online]. Available: http://dx.doi.org/10.1016/j.advengsoft.2016.01.008