A Standardized Benchmark Set of Clustering Problem Instances for Comparing Black-Box Optimizers

Diederick Vermetten
Sorbonne Université, CNRS, LIP6
Paris, France
Diederick.Vermetten@lip6.fr

Catalin-Viorel Dinu
Sorbonne Université, CNRS, LIP6
Paris, France
Catalin-Viorel.Dinu@lip6.fr

Marcus Gallagher
School of EECS, University of Queensland
Brisbane, Australia
marcusg@uq.edu.au

ABSTRACT

One key challenge in optimization is the selection of a suitable set of benchmark problems. A common goal is to find functions which are representative of a class of real-world optimization problems in order to ensure findings on the benchmarks will translate to relevant problem domains. While some problem characteristics are well-covered by popular benchmarking suites, others are often overlooked. One example of such a problem characteristic is permutation invariance, where the search space consists of a set of symmetrical search regions. This type of problem occurs e.g. when a set of solutions has to be found, but the ordering within this set does not matter. The data clustering problem, often seen in machine learning contexts, is a clear example of such an optimization landscape, and has thus been proposed as a base from which optimization benchmarks can be created. In addition to the symmetry aspect, these clustering problems also contain potential regions of neutrality, which can provide an additional challenge to optimization algorithms.

In this paper, we present a standardized benchmark suite for the evaluation of continuous black-box optimization algorithms, based on data clustering problems. To gain insight into the diversity of the benchmark set, both internally and in comparison to existing suites, we perform a benchmarking study of a set of modular CMA-ES configurations, as well as an analysis using exploratory landscape analysis. Our benchmark set is open-source and integrated with the IOHprofiler benchmarking framework to encourage its use in future research.

CCS CONCEPTS

• **Theory of computation** → **Design and analysis of algorithms**; **Bio-inspired optimization**.

ACM Reference Format:
Diederick Vermetten, Catalin-Viorel Dinu, and Marcus Gallagher. 2025. A Standardized Benchmark Set of Clustering Problem Instances for Comparing Black-Box Optimizers. In *Foundations of Genetic Algorithms XVIII (FOGA '25), August 27–29, 2025, Leiden, Netherlands.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3729878.3746699

FOGA '25, August 27–29, 2025, Leiden, Netherlands

ACM ISBN 979-8-4007-1859-5/2025/08
https://doi.org/10.1145/3729878.3746699

1 INTRODUCTION

Benchmarking is a key aspect in the development and analysis of optimization algorithms. Through rigorous benchmarking, we can gain insights into the relative strengths and weaknesses of different algorithmic ideas when applied to certain types of challenge, understand optimization behavior, and much more. Generally, we aim to benchmark either on a set of problems with well-understood structure, since this can lead to useful insight about algorithm behavior, or on problems which are assumed to match some aspect of specific real-world problem domains.

It is not always straightforward to find benchmark problems which meet this criterion of representing specific types of challenges. While some aspects are well-covered by a variety of benchmarking suites, others might be much more sparsely distributed, and thus more difficult to analyze in detail.

One such aspect that has been little studied is permutation invariance, which occurs for example when the 'true' search should take place on a set of items, but is represented as an ordered list. This happens when the ordering between variables (or sets of variables) doesn't impact the evaluation function. In this case, there are natural symmetry regions in the space, which correspond to the same underlying solution space.

Problems with permutation invariance exist in a wide variety of domains, ranging from facility location[9], placement problems[8], clustering[16] and neural network training[11, 23] to design optimization problems [32]. In statistics and machine learning, this invariance is referred to as the label switching problem, for example in mixture models [31, 38].

There is a large amount of previous work where clustering problem instances have been used to evaluate the performance of a range of different types of global, black-box, metaheuristic and other optimization algorithms. Unfortunately, it is typically difficult to perform a meta-analysis or comparison of previous results across papers, because of issues such as different experimental settings, different performance metrics reported and different datasets utilized[6, 28]. Providing standardized sets of problem instances, making data and software available and integration with benchmarking tools is critical to support the research community in producing more comparable experimental results. In this paper, we introduce a benchmark suite based on these clustering problems which is integrated with the IOHprofiler framework[13, 44] for usability. We also provide an interface to create custom clustering problems and analyze our proposed suite by performing several benchmarking studies on it and comparing it to COCO's well-known BBOB problem suite [20, 21].

2 THE CLUSTERING PROBLEMS

To create our benchmarking suite, we make use of (centroid) data-clustering problems, which are commonly seen in machine learning and data analysis[24]. We build on previous studies which showcase the potential use of these types of problems as optimization benchmarks [16, 37]. The Mean Squared Error (MSE) clustering problem can be defined as follows. Given a training set of n data points, $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subseteq \mathbb{R}^d$, determine k cluster centers $C = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k\} \in \mathbb{R}^d$ to minimize:

$$f(C \mid X) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} b_{i,j} ||\mathbf{x}_i - \mathbf{c}_j||^2 \tag{1}$$

where

$$b_{i,j} = \begin{cases} 1 & \text{if } ||\mathbf{x}_i - \mathbf{c}_j|| = min_j ||\mathbf{x}_i - \mathbf{c}_j|| \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note that the decision variables of the problem, $\mathbf{y}$ are the d-dimensional coordinates of each cluster center, collected into a vector, meaning that the optimisation problem is of dimensionality $m \equiv kd$

$$\mathbf{y} = (y_1, \ldots, y_m) \equiv \mathbf{c} = (c_1, \ldots, c_{kd}). \tag{3}$$

The key parameters of this problem class are the dimensionality (d), cardinality ($n = |\mathcal{D}|$) and distribution of the data points and the number of cluster centers (k). In general, this class of problems is known to be NP-hard[1].

For $k = 1$, the fitness landscape is a (convex) quadratic bowl, but with higher values for k, the landscape becomes a combination of quadratic bowls of different sizes, intersecting with discontinuous ridges[16]. Consequently, the landscapes of clustering problems are in general non-linear and non-convex with a large number of local optima.

Using MSE implies that the (squared) Euclidean distance measure is used. Since the choice of distance and error measure could potentially have an impact on the resulting optimization landscape, we provide a flexible problem generator where these design choices can be modified in the future. However, we stick with these default values for the provided benchmark suite and for any experiments presented in this paper.

Given this definition, we can create instances of clustering problem by defining datasets $\mathcal{D}$ and a corresponding number of cluster centers to find k. In Figure 1, we show the contour plots of fitness landscapes for two datasets with $d = 1$ and $k = 2$.

2.1 Symmetries

The clustering problem is invariant under permutations of the cluster centers. That is, reordering the index of the cluster centers does not change the objective value, as the assignments of data points depend solely on their distances to the centers, not on the index itself. Formally, for any permutation π of the set $\{1, \ldots, k\}$, the permuted set of centers $C' = \{\mathbf{c}_{\pi(1)}, \mathbf{c}_{\pi(2)}, ..., \mathbf{c}_{\pi(k)}\}$ satisfies $f(C' \mid X) = f(C \mid X)$. As a consequence, the optimization problem has $k!$ symmetric global optima that are functionally equivalent but differ only in the ordering of the cluster centers. This behaviour also translates to the search space when we concatenate C into $\mathbf{y}$. These symmetries partition the search space into equivalent regions, each corresponding to a unique permutation of the cluster centers. In the case of $k = 2$, for example, the symmetry is reflected as a diagonal ridge dividing the search space into two congruent regions, as illustrated in Figure 1. This type of symmetry is also found in some combinatorial problems, such as graph coloring [39].

The question of symmetry is present in a wide variety of optimization problems, as discussed earlier. In machine learning, the symmetry is often ignored since there is no explicit use for labelling the latent variables and doing so allows for easy formulation of optimization (e.g. gradient-based techniques in an unconstrained space). Different representations can be used that avoid the symmetry, such as convex relaxations [4] or semidefinite programming [33].

There has also been some interest in modifying optimization algorithms to handle this specific challenge. In Bayesian optimization, recent studies have proposed modifications of the kernel function to account for the permutation-symmetries present in these search spaces, resulting in promising improvements in performance [10, 14]. In the evolutionary algorithms community, questions of invariance in the search space have focused on transformations such as rotation and translation [19], but the question of permutation invariance has not been studied as broadly.

2.2 Neutrality

Another important property of clustering problems is that the fitness landscapes contain a significant amount of neutrality. Neutrality has been studied in fitness landscape analysis and across a number of problem domains, predominantly in the discrete case (see, e.g. [40] and the references therein). If an optimization problem is formulated with one or more decision variables that have no/negligible effect on the objective function, this will create a search space with neutrality. Therefore, it seems reasonable to expect that this is a property of some continuous real-world optimization problems.

As formulated above, the clustering problem is unconstrained, but the range of each data variable provides a soft boundary constraint that can be used to initialize an algorithm, since a solution will not be improved by moving cluster center outside of the range of the data. If one (or more) cluster centers, $\mathbf{c}_j$ is not the closest cluster center for any of the data points $\mathbf{x}_i$ (i.e. for $\mathbf{c}_j$, $b_{i,j} = 0 \ \forall \ i \in \{1, \ldots, n\}$) it means that the landscape contains regions that are perfectly flat. This can be seen in the two rightmost plots of Figure 1, where the search ranges have been extended beyond the original data points, revealing regions of neutrality.

Similarly to the symmetry property, additional conditions can be added to the problem formulation to reduce neutrality, such as requiring each cluster center to be closest to at least one data point and that each data point is closest to only one cluster center (i.e. non-equidistant), see, e.g.[5]. However, this would require constraint handling mechanisms to be incorporated into the optimization algorithms, creating more complexity in the algorithm configuration.

2.3 Benchmark Suite

Clustering is a widely-studied problem, and a wide variety of different datasets have been utilized to evaluate the performance of clustering algorithms (e.g.[15]). Much of this work is not focused

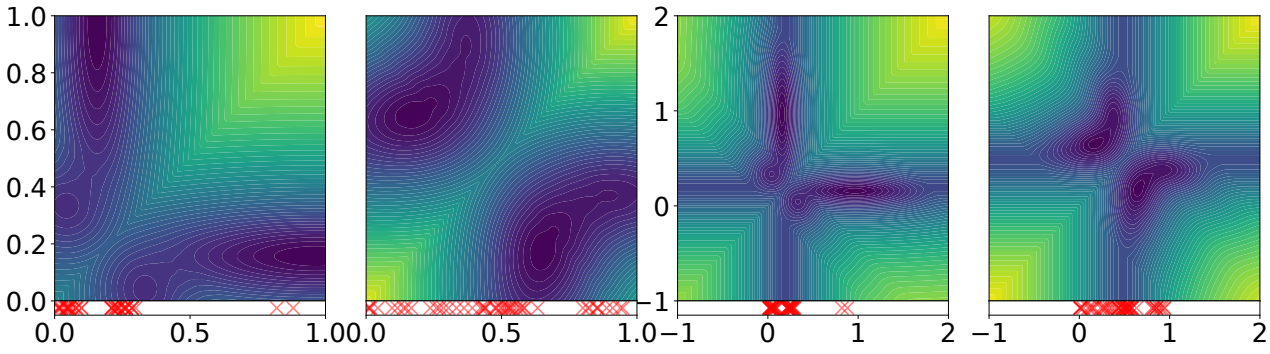


Figure 1: Contour plots of selected clustering problems based on 1-dimensional data (visualized as red crosses below the respective plots) with two cluster centers. The two right-most plots are based on the same data as the two left-most plots, but plotted using a wider domain.

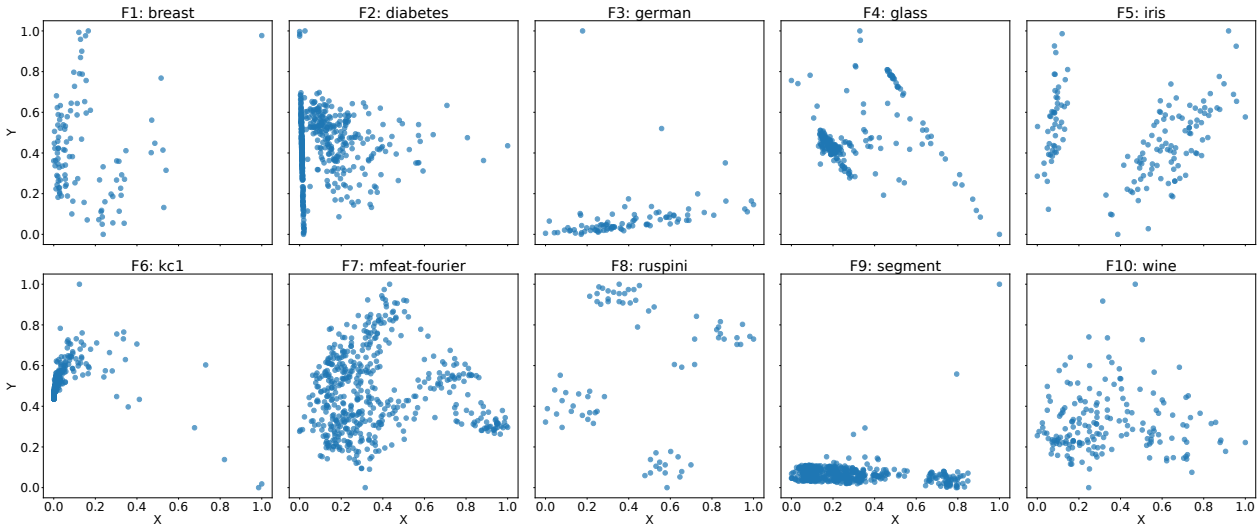


Figure 2: Data underlying each of the 10 problem sets, where the original datasets have been projected onto the space spanned by the first two principal components.

on evaluating the quality of the solutions found in terms of the optimization objective function, but rather uses metrics to evaluate the quality of the data clustering produced, as well as the performance of using the clustering as a classifier (assuming the data contains labels for true classes). Many papers do not report the objective function values found, instead focusing on assessing qualities of the clustering such as the separation of points between clusters or using labelled datasets to treat the clustering as a classifier. Unfortunately, this limits the ability to assess the performance of the underlying optimizer on the objective function.

A set of 27 problem instances was produced and used in [16], with Matlab code and information such as (approximate) global optimum solution vectors and objective values available at: https://marcusgal.github.io/ess_clustering.html. However, one limitation is that the problem instances vary over many dimensionalities, while it is often useful to compare optimizers on a set of problems of a fixed dimension. Two of these problem instances were also included in the MLDA [26, 37], incorporated into Nevergrad [7].

To create our benchmark suite, we select a total of 10 datasets from different machine learning context. Three of these (German towns, Ruspini and Iris) are selected because they have been studied in this data-clustering context in previous work [16]. The remaining 7 datasets have been taken from the popular UCI-collection [3], where we focused on the most popular datasets with numerical features. Since the fitness-calculation scales with the number of samples, we limit the number of data points to 5 000.

Since we want to ensure straightforward aggregations over different problems in our suite, we opt to ensure each dataset has the same number of features (dimensionality). This is achieved by using principal component analysis on each of the selected datasets and keeping only the first two principal components. Since we opt

to use $k \in \{2, 3, 5, 10\}$, this leads to problem dimensionalities of $n \in \{4, 6, 10, 20\}$ for each of the 10 datasets.

To provide consistent search domains for the optimization algorithms, we min-max normalize the data-space (after PCA) to $[0, 1]^n$. The resulting datasets are visualized in Figure 2. These figures show that there is quite some variety between the datasets in terms of the density of datapoints and the presence of outliers.

It is important to note that we treat these datasets purely as data for the data clustering problem, and thus any information about potential 'true' labellings has been removed. This differs from works that look at clustering from a machine learning context (e.g., [15]), where labels are used as ground truth. While this means we don't have known optimal solutions for our problem suite, it provides the required flexibility to e.g. modify the number of cluster centers.

2.4 IOHClustering Python Package

To ensure the accessibility of our proposed benchmark suite, we created a new Python package 'IOHclustering', which integrates with the existing IOHprofiler framework. This allows users to rely on the logging methodology from IOHexperimenter [13], and subsequently analyze and visualize their performance data using tools such as IOHanalyzer [44] and IOHinspector [42].

In addition to the suite of problems described above, IOHclustering also integrates a generic way to instantiate arbitrary clustering problems by providing a dataset and a number of cluster centers. The resulting problem can be further customized by changing the distance metric used for cluster assignment and/or the error metric used to judge the cluster quality. As such, new instances can be generated from arbitrary combinations of these four parameters.

3 ANALYSIS OF BENCHMARK SUITE

When introducing a new problem suite, it is important to consider its usefulness for benchmarking tasks. To this end, we consider the desirable properties for benchmarking problem sets as described in [6], which are as follows:

- Diversity: the problems in the suite should be sufficiently varied, ideally from both an algorithm performance perspective (easy / hard) and from a more fundamental problem characteristics perspective (different kinds of optimization challenges).
- Representativeness: the problems in the suite should contain challenges which are likely present in a set of real-world optimization problems.
- Scalability and Tunability: Ideally, a problem suite / framework should make it possible to tune the characteristics of the problems, whether on a high level (e.g. number of variables) or on a lower level (e.g. variable interactions, degree of multimodality)
- Known solutions: If the global optima of the problems are known, it makes it easier to compare performance of the algorithms. If no exact values are known, strong best-known results could be useful alternatives.

When considering these criteria, we note that the question of representativeness is the driving motivation for this suite. As discussed in Section 2, the inherent symmetry is a problem characteristic which occurs in a variety of domains, but is not widely covered by optimization benchmarks.

The question of scalability is incorporated into the problem suite in the form of the number of clusters k, which directly impacts the number of variables of the optimization problem. Note that this is a different type of scalability as commonly found in e.g. BBOB, since we don't have any guarantees that the optimization landscape for a given dataset and k value is similar to the landscape for the same dataset with changed k.

Regarding tunability, we should note that a benchmark suite of fixed size will necessarily be limited in this regard. However, through providing an accompanying problem generator, we hope to facilitate the creation of clustering problems with more flexible problem characteristics. Some work has considered the idea of searching the problem space by modifying points (or properties of their distribution) to create problems that are more challenging, or provide differentiation between different algorithms[17, 18].

While our problem suite does not include known optima, we can find very strong baseline values for solution quality. Since we are dealing with clustering problems, we can get these performance baselines by running the well-known (non-black-box) *k-Means* algorithm, more specifically with *K-Means++* initialization as used in many software packages by default[2]. Since this algorithm is stochastic, we perform 100 repetitions on each problem in our suite to collect baselines. For ease-of-use, these values are also accessible in our Python package.

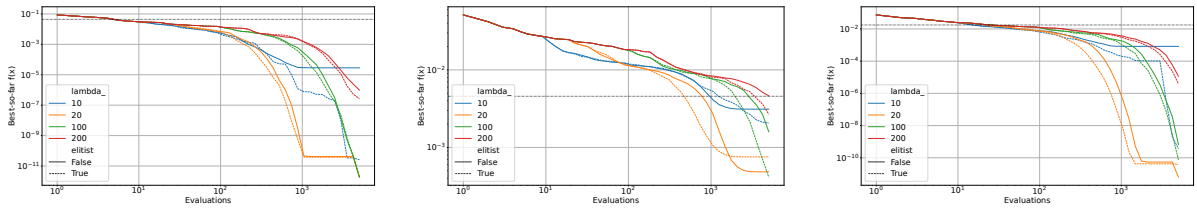
3.1 Benchmarking Study

While the previous criteria for benchmark suite quality were relatively straightforward to address, the question of problem diversity requires a more involved experimental setup. We can judge diversity in different ways: diversity within the benchmark suite itself, and diversity with respect to existing benchmark suites. Subsequently, this diversity can be further split into diversity of algorithm performance and diversity of problem characteristics. We will address each of these aspects, starting with algorithm performance within the suite. Critically, we don't attempt to find the best type of algorithm for this type of benchmark problem, but rather focus on exploring the properties of the proposed suite through a set of different benchmarking experiments.

To analyse the diversity of algorithm performance, we run a benchmark study using a large set of configurations of the *CMA-ES* algorithm [22]. We make use of the Modular *CMA-ES* framework [12] for this purpose, and create a set of 128 configurations by modifying the following parameters (leaving the rest as default in the modcma package):

- Covariance adaptation: On / Off
- Elitism: On/ Off
- Boundary Correction: Off / Saturate
- Lambda: 5, 10, 20, 100, 200
- Mu: 5, 10, 20, 50, 100

Note that while the full enumeration of these settings yields a slightly larger set of configurations, we require $\lambda \geq \mu$ to have a valid



(a) Function 1, k=2 (b) Function 1, k=10 (c) Function 2, k=3

Figure 3: Some example convergence curves (geometric mean) for different population sizes and elitism options (bound correction off, covariance on, $\mu = 10$). All values are differences from the best value found by 100 repetitions of *K-Means++*.

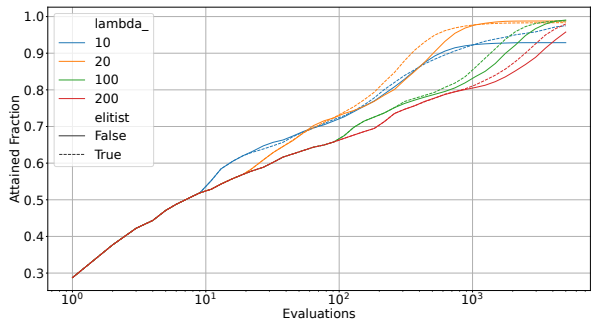


Figure 4: EAF over all 10-dimensional problems in our suite, for different population sizes and elitism options (bound correction off, covariance on, $\mu = 10$). The EAF bounds are set to the *K-Means++* baseline and the worst seen value by all *CMA-ES* variants, respectively (with a log-scaling between them).

algorithm. These five parameters were chosen not to try to achieve optimal algorithm performance, but to cover a range of different algorithmic behavior of the resulting *CMA-ES* configurations.

Each of the 128 selected *CMA-ES* configurations is run on each of the 40 benchmark problems, with 25 independent repetitions of budget 5000 (not scaled by problem dimensionality). We ensured that random seeds were set equally between configurations to ensure all configurations with a given population size start from the same set of samples (but with independent seeds for the different runs).

To ensure reproducibility of this benchmarking setup, and all other experiments discussed in this paper, we provide a Zenodo repository which contains the script, data and visualization methods required to reproduce all presented results, as well as several additional figures [41].

First, we analyze the convergence behavior of a smaller set of *CMA-ES* configurations. Figure 3 shows the convergence curves for 3 different benchmark problems, where for each problem, 8 configurations are selected by varying λ and elitism. The equivalent convergence curves for all other problems are available in our Zenodo repository [41] (for both the original and relative function values). From these convergence curves, we observe that for low-dimensional problems, the total budget provided is sufficient for all selected *CMA-ES* configurations to converge to within 10^{-5} of the best *K-Means++* performance value. However, differences between configurations are clearly present in terms of convergence speed, with the lower population sizes resulting in notably faster convergence. However, as dimensionality increases, differences between configurations become more pronounced and lower values of λ become more prone to stagnation.

In addition to the convergence curves, the fact that we have baseline performance values from *K-Means++* means that we can create aggregations over different datasets. We can do this in the form of the Empirical Attainment Function (EAF), with its bounds chosen to be the *K-Means++* value and the worst-observed value in all *CMA-ES* runs respectively. Note that this corresponds to the ECDF between these bounds if we assume an infinite number of targets [29]. The EAF aggregated over all Datasets with $k = 5$ is shown in Figure 4.

One clear result from analyzing the EAF in Figure 4 (and the convergence curves in Figure 3) is the impact of the overall budget on the relative ranking of different configurations. While the remaining analysis of algorithm performance focuses on the fixed-budget perspective (final performance after 5000 evaluations), this serves as a proof-of-concept to illustrate the proposed benchmark problems, not as a judgment for which algorithm configuration is the most appropriate for this suite.

To better understand whether our benchmark suite is able to distinguish between algorithms, we zoom in on the overall differences in performance between all algorithms in our portfolio. In particular, we compare the performance of the best and worst algorithm configurations to gauge the breadth of performance found on each problem. In Figure 5 we show these differences for each problem in our proposed suite. From this figure, we can see that the main factor which determines the scale of performance differences is the problem dimensionality. However, even for the setting with dimensionality 4, there are still differences present between the best and worst versions of the *CMA-ES*. When comparing differences between functions of the same dimensionality, we observe there are also some visible patterns, with the relative ordering of the performance differences remaining relatively stable across dimensionalities.

Next, we look for patterns in the top-ranking configurations on each benchmark problem. To this end, we create a stacked bar chart of module counts in the top 8 configurations on each problem. In Figure 6 we show these results aggregated by dataset. From this, we

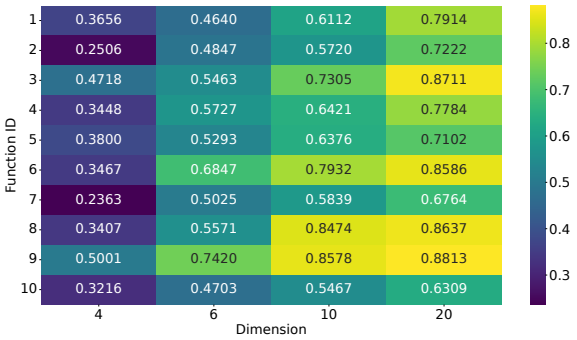


Figure 5: Relative differences (in terms of function average final function value) between best and worst *CMA-ES* configurations in the portfolio.

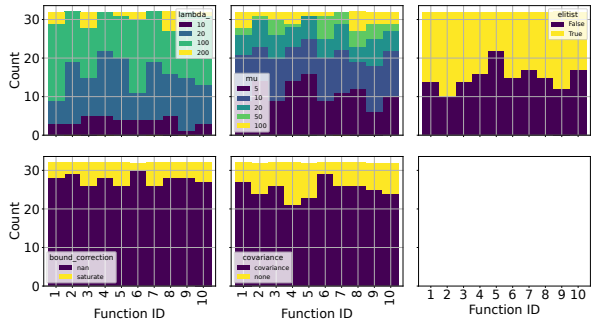


Figure 6: Distribution of parameters in the top 8 (out of 128) configurations on each function, grouped by dataset.

see that the overall differences between datasets are relatively small, with similar distributions of best-performing parameter settings. While the choice of elitism seems to be rather evenly split between top performing configurations, both the boundary correction and covariance modules have a setting with is much more common than its alternative.

For boundary correction, most top-performing configurations avoid using the saturation mechanism; instead, they ignore the boundaries entirely. This suggests that the regions outside of the box spanned by the datapoints might still contain useful information for the search process. For the covariance parameter, it is most commonly enabled, indicating that there might be some interactions between variables for which adapting the covariance matrix is beneficial.

Finally, we compare the performance of our optimization algorithms to the baseline values found by *K-Means++*. Here, we focus on the impact of population size, by taking the best-performing configuration for each population size and showing the absolute deviation to the K-means result. These differences are shown in Figure 7, where we notice that *CMA-ES* reaches values which are very close to those found by K-means. This seems to confirm that our earlier observations on consistent convergence in low dimensionality corresponds to the (likely) global optimal solution. This seems

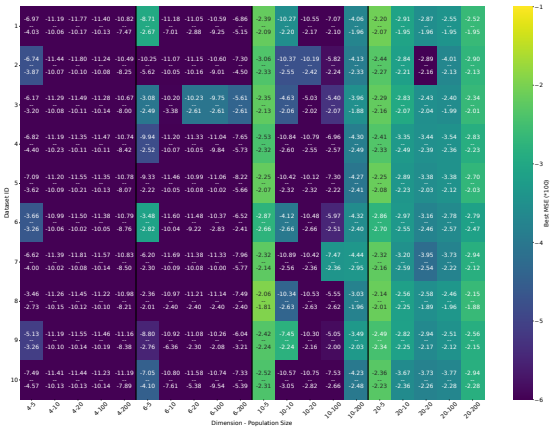


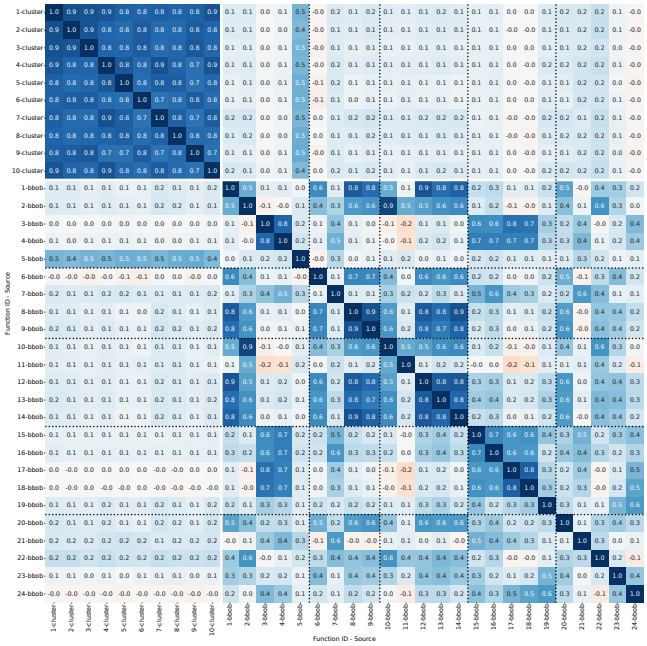
Figure 7: For each problem, the performance found by the best *CMA-ES* configuration with the specified λ value, shown as difference to the minimal value found by 100 repetitions of *K-Means++*. Performance is defined here as $log_{10}(f(x))$ with $f(x)$ being the final function value reached after the full budget is exhausted. The colors correspond to median, while the numbers correspond to the 0.1 and 0.9 quantiles respectively

to indicate that the CMA-ES can serve as a rather effective baseline algorithm for future comparison of optimization algorithms on this benchmark suite, while still showing potential for further refinement.

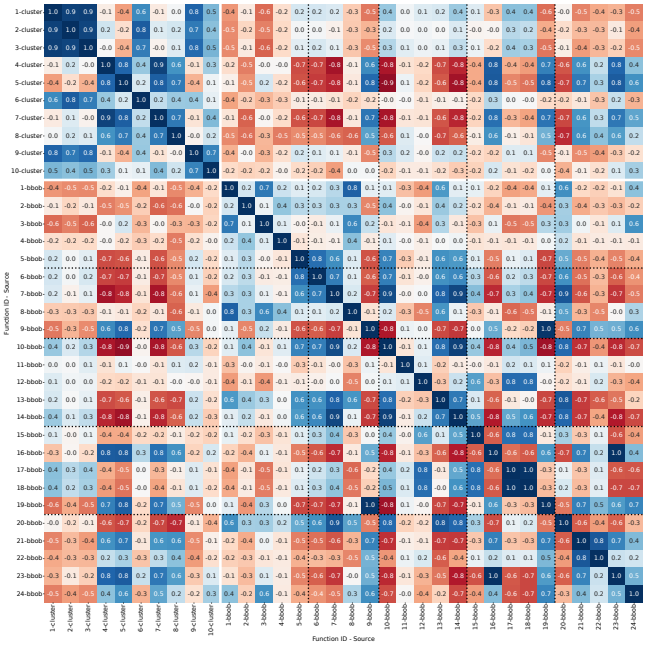
3.2 Comparison to BBOB

In order to understand whether our proposed benchmark suite does indeed result in different algorithmic challenges than existing benchmarking suites, we compare our problems to the equivalent-dimensional problems from the BBOB suite. We identify two main axes along which we can assess problem similarity: algorithm performance and landscape characteristics. From the algorithm performance perspective, we utilize the relative performance rankings of the CMA-ES configurations from Section 3.1, and compare this to their equivalent ranking on COCO's BBOB suite [20] (where we thus use problem dimensionalities $\{4, 6, 10, 20\}$). While the BBOB suite has an instance generation mechanism [21], this is not present in our suite, and thus we stick with only the first instance as a representative of each BBOB function.

For each problem, we create the ranking of the 128 *CMA-ES* configurations based on the best average final fitness value found. Based on these rankings, we calculate Kendall's τ correlation [25] for each pair of problems, and visualize these correlations in Figure 8a. In this figure, the clustering problems are clearly separated from the BBOB functions, with high correlation of the CMA-ES configuration rankings between all 10 clustering problems. However, the correlations to the BBOB problems are generally much lower, with a peak for F_5, the linear slope. This could point to two underlying factors: on the one hand, more exploitative versions of *CMA-ES* tend to work well on the linear slope, and based on the results presented in Figure 7, the same seems to be true for the



(a) Kendall's τ correlation based on performance (best-so-far) of all 128 modCMA configurations.



(b) Cosine similarity between (here: standardized to 0 mean, variance 1; [0,1] normalized version available as well) ELA feature vectors (lin-model, meta, distr, level, pca, nbc and ic sets).

Figure 8: Pairwise similarities between the clustering and BBOB problems. Shown here for dimensionality 10, other dimensionalities are available in our Zenodo repository [41].

clustering problems. However, since we don't observe the same high correlation to F_1, it might also be related to the fact that the boundary correction method is generally disadvantageous for the linear slope, which, according to Figure 6, is also the case on the clustering problems.

Overall, Figure 8a shows that the clustering problems have high internal problem similarity, but differ from most of the problems present in BBOB. However, similarity in algorithm performance does not necessarily imply similarity in problem structure. To further asses the landscape differences between these problem sets, we can employ exploratory landscape analysis (ELA) [30]. To collect ELA features, we generate a set of 4096 samples in the search space (using a Sobol sampler). We then min-max normalize the function values to $[0, 1]$ as recommended in [35], use PFlacco [36] to calculate the ELA features from the commonly used feature sets (linear model, meta, distribution, PCA, nearest best clustering, information content; as used in e.g. [27]) and normalize the resulting feature values (mean 0, standard deviation 1). Notably, we don't perform feature selection (except for the standard removal of feature-set calculation times).

Based on the normalized ELA features, we use cosine similarity on each pair of problems. The resulting heatmap is shown in Figure 8b. From this figure, we notice that the similarities between the set of clustering problems are still generally positive, but show more diversity. The similarity between clustering and BBOB problems gives more evidence for the claim that these problem sets are quite different, with most values moderately negative. There are outlying values for some of the more multi-modal problems, which could be expected in this problem dimensionality, as we have at least $5! = 120$ equivalent global minima (since we have $k = 5$ cluster centers). When considering other problem dimensionalities (and thus differing numbers of symmetries), as we make available in our Zenodo repository[41], we observe that this observation becomes stronger for larger k, while the lower k-values show less clear patterns in terms of which BBOB-functions are more similar to our clustering problems.

3.3 Analysis of Optimization Landscapes

To gain a deeper understanding of the landscapes induced by our clustering problems, we investigate the property of multi-modality. While the presence of multi-modality arising from problem symmetries is immediately apparent, it remains unclear whether each symmetry region contains only one or multiple local minima. To address this question, we use several local optimization algorithms, each initialized from 50 distinct points within the search space. These initial points are chosen such that they lie within the same symmetry region (enforced by ordering the first coordinate of the cluster centers).

We consider the following local search methods:

- Powell's method ([34], implementation from scipy [43]),
- L-BFGS-B ([45], implementation from scipy [43]),

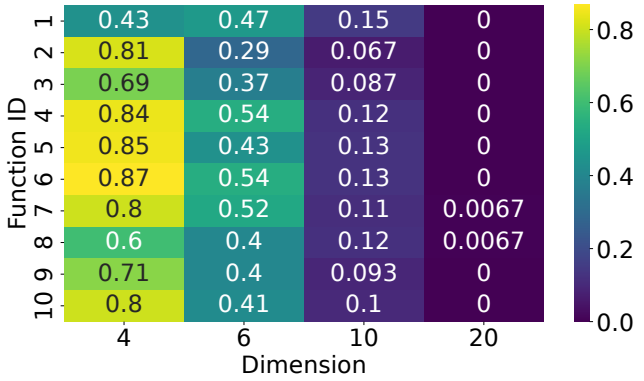


Figure 9: Fraction of solutions that ended up in the same region of symmetry as the starting point of the local search.

- *(1+1)-CMA-ES* (discussed before, implementation from [12]), with an initial step size $\sigma_0 = 0.1$ to promote local search behavior.

In addition to probing the multi-modal structure, this setup allows us to examine the extent to which local search methods remain confined within their initial symmetry regions. This is done by ensuring the search starts with a solution where the first components of each cluster center are ordered, and checking whether this is still the case for the final found solution. Figure 9 reports the proportion of solutions that remained within their original regions throughout the optimization process. As the dimensionality of the problem increases, the diversity of the final solutions also increases. In low-dimensional settings, such as for $k = 2$, approximately 75% of the solutions remain within the same symmetry region as their initial starting points after local search. However, as the dimensionality grows, this proportion decreases significantly. At 10 dimensions, only around 11% of the solutions remain within their initial region, and at 20 dimensions, virtually no solutions are found to stay within their original symmetry regions.

Beyond identifying local minima, we aim to assess the diversity of solutions and the prevalence of suboptimal local minima, providing deeper insights into the structure and multimodality of the landscape. To this end, we proceed as follows:

(1) Aggregate all final solutions obtained from the three local search algorithms.
(2) Perform hill-valley tests (with a number of 4 intermediary points) between all pairs of solutions to detect the presence of intermediate minima.
(3) Construct a solution network, where nodes represent solutions and edges indicate the existence of an intermediate minimum between them.
(4) Identify fully connected subgraphs (cliques) within this network to group structurally similar solutions.
(5) Select representative solutions (those with the best objective value) from each clique.

After identifying the representative solutions of each clique, we perform a pairwise comparison to investigate whether the associated clusters originate from the same basin of attraction. For each pair of representatives, we proceed as follows: starting from the representative with the worse objective value, we perform a simple local search by sequentially generating perturbed points in its vicinity. At each iteration, small random perturbations are applied, and the new point is accepted if it improves the objective value. This process continues until either (i) the search point moves sufficiently close to the better representative in the search space, suggesting that the two solutions belong to the same basin, or (ii) no further improvements can be found through perturbations, indicating that the solutions likely reside in different basins. This procedure enables us to characterize not only the diversity of the local minima but also the connectivity between basins, offering deeper insight into the ruggedness and structure of the underlying search landscape.

We applied the previously described procedure to the final solutions obtained by the local search algorithms. As illustrated in Figure 10c, the results reveal a high degree of solution diversity across problem instances and dimensions. Some problems yield a single solution across runs, while others produce multiple distinct solutions, indicating the presence of several basins of attraction. These findings offer some evidence of multi-modality in the underlying landscapes.

Moreover, we observe that the number of basins increases with problem dimensionality, which matches the intuition that higher-dimensional landscapes become more rugged and fragmented. This trend is also visible in Figures 10a and 10b, where we present the solutions found on the Ruspini dataset (function F8) for $k = 2$ and $k = 3$ clusters, respectively. For $k = 2$ (Figure 10a), two distinct solutions are identified, while for $k = 3$ (Figure 10b), four unique solutions emerge.

3.4 Breaking the Symmetry

As a final perspective on clustering problems for optimization algorithms, we focus in more detail on their inherent symmetry, as defined in Section 2.1. To illustrate these symmetries, we use a simple 1-dimensional dataset with $k = 2$, to have a search-space with 2 dimensions and 2 regions of symmetry[16, 18] (so $f(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}) = f(x_{1,0}, x_{1,1}, x_{0,0}, x_{0,1})$ for all $\vec{x}_0, \vec{x}_1 \in \mathcal{R}^2$). Since symmetry regions imply multi-modality, this could potentially hinder the performance of an optimization algorithm, especially when a population-based algorithm starts the search in multiple symmetrical regions, potentially leading to redundant search steps.

Because of this, it could be beneficial to find a representation of this problem which removes the symmetry. In essence, this would involve creating a (bijective) transformation function $t : [0,1]^m \rightarrow S \subset [0,1]^m$ where S is a single region of symmetry. The problem of finding such a transformation has commonalities with the problem of sampling in a unit-simplex, for which the Dirichlet distribution can be used. As such, we can make use of a modified stick-breaking procedure:

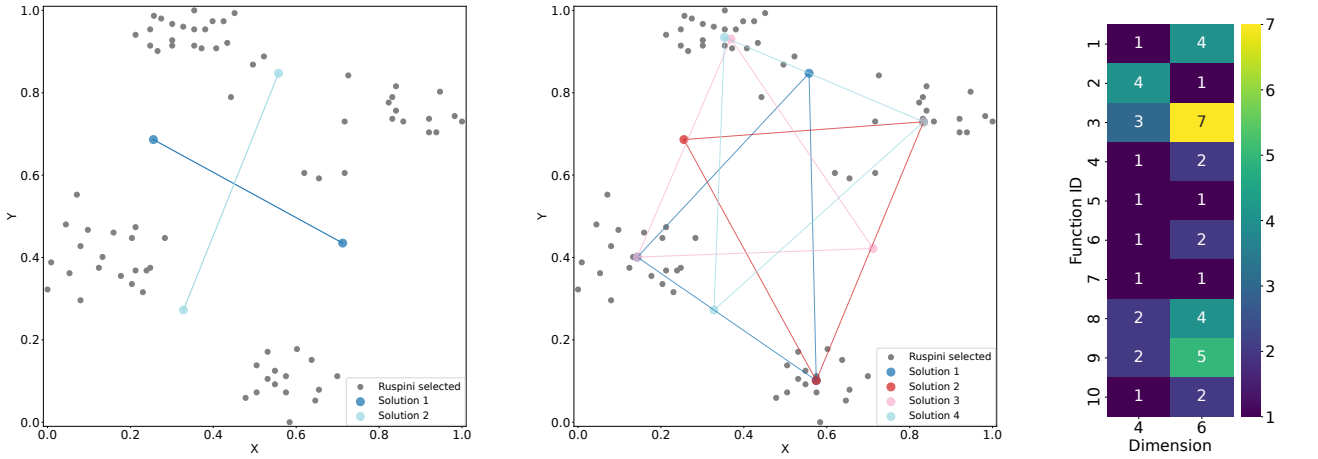
We let

$$t(x_1) = F^{-1}_{\text{Beta}}(x_0 \mid 1, X)$$

and

$$t(x_i) = t(x_{i-1}) + (1 - t(x_{i-1})) \cdot F^{-1}_{\text{Beta}}(u_i \mid 1, X - i)$$

for each $i \in \{2 \dots k\}$. Here, F^{-1}_{Beta} refers to the inverse CDF of the Beta-distribution with $\beta = X$.



(a) Distinct solutions found by local search algorithms on function F8 with $k = 2$.

(b) Distinct solutions found by local search algorithms on function F8 with $k = 3$.

(c) The number of distinct solutions found by local search algorithms.

Figure 10: Number of distinct local minima discovered by local search algorithms across different problem instances. The examples for $k = 2$ and $k = 3$ demonstrate cases where multiple solutions were found, indicating the presence of multiple basins of attraction even for small numbers of clusters.

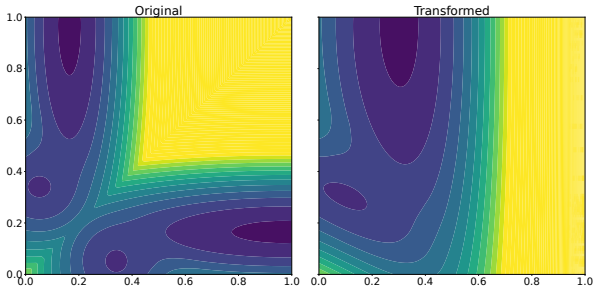


Figure 11: Example of Original and Transformed version of a 1-dimensional clustering problem with $k = 2$.

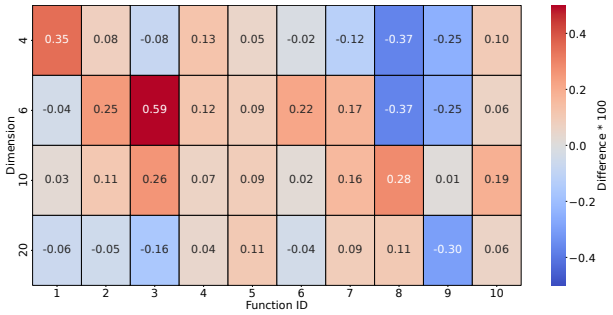


Figure 12: Differences in mean performance of *CMA-ES* on the original vs transformed space. Negative values correspond to functions where the original representation led to better performance.

To illustrate this transformation on our 2-dimensional search space, we show both the original and transformed version of a clustering problem (with $d = 1, m = k = 2$) in Figure 11. We note that to generalize this transformation to higher values of d, the transformation can be applied to the first component of each cluster center, while leaving the remaining coordinates intact.

To verify whether this transformation has a positive impact on the performance of optimization algorithms, we benchmark the default configuration of *CMA-ES*, both with and without this transformation, on each of the 40 problems in our proposed suite. We perform 25 repetitions on each problem, and compare the final function value found between the two setups. Their relative differences are visualized in Figure 12.

We can see from Figure 12 that our proposed transformation results in similar performance to that of the default problem representation. However, there are some problems where the performance becomes worse, which might to suggest that these transformed search spaces become harder to navigate for this algorithm. When looking at the definition of the transformation, we should note that the first transformed variable depends only on x_0, but the second depends on both x_1 (for the Beta-distribution) and t_0 (for the weighting). Generally, each variable in the transformed space depends on the variables before it, inducing a hierarchy to the search variables. This hierarchical nature is inherently challenging for most population-based algorithms, so we can conclude that our proposed transformation does not necessarily result in an easier to optimize representation for the clustering problem.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed and analyzed a suite of optimization problems based on data clustering. These problems provide optimization challenges not commonly addressed in existing problem

suites, but which are present in a variety of real-world domains. By performing a selection of benchmarking studies, we showcased the diversity of our problem set in relation to the commonly used BBOB suite, both from an algorithmic performance and landscape characteristic perspective. It is however still an open question how this compares to more practical optimization problems with similar symmetry characteristics in different real-world domains.

With the introduction of this problem suite, and its integration with the IOHprofiler framework, we aim to make it easier for other researchers to utilise a wider variety of problems for algorithm benchmarking as well as promote further research on problems with neutrality and permutation symmetries.

Since this paper focused on the introduction and analysis of the problem suite, it does not present a complete picture of the state-of-the-art methods for dealing with these challenges. Further research and benchmarking efforts could help identify the algorithmic ideas which would most effectively address the challenges present in our proposed problem suite. While our preliminary experiments showcase some potential benefits a symmetry-breaking transformation, further systematic research in this direction is required to determine the extent of its usefulness. Additionally, more extensive investigations into alternative ways of breaking the symmetry would be highly useful, as could be done e.g. by introducing constraint mechanisms or different mappings to the unit simplex.

Since more clustering problem instances can easily be specified with additional datasets and scaled to higher dimensionalites by increasing k, we, in addition to the problem suite, also provide a simple problem generator. Since this generator comes with increased flexibility in terms of the clustering setup, it would be especially interesting to analyze how changes to the setup, e.g. changing the error measure to a maximum error per cluster center, would impact the fundamental landscape properties of the resulting optimization problem, and in turn how this impacts the relative performance of different types of optimization algorithms. This could help fine-tune algorithms to address challenges in specific problem domains.

ACKNOWLEDGMENTS

Parts of this work were performed while Diederick Vermetten was employed at LIACS, Leiden University. This project was partially supported by the European Union (ERC, "dynaBBO", grant no. 101125586). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. 2009. NP-hardness of Euclidean sum-of-squares clustering. *Machine learning* 75 (2009), 245–248.
[2] David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding*. Technical Report. Stanford.
[3] Arthur Asuncion, David Newman, et al. 2007. UCI machine learning repository.
[4] Pranjal Awasthi, Afonso S Bandeira, Moses Charikar, Ravishankar Krishnaswamy, Soledad Villar, and Rachel Ward. 2015. Relax, no need to round: Integrality of clustering formulations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*. 191–200.
[5] Adil M Bagirov. 2008. Modified global k-means algorithm for minimum sum-of-squares clustering problems. *Pattern Recognition* 41, 10 (2008), 3192–3199.
[6] Thomas Bartz-Beielstein, Carola Doerr, Daan van den Berg, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, William La Cava, Manuel Lopez-Ibanez, et al. 2020. Benchmarking in optimization: Best practice and open issues. *arXiv preprint arXiv:2007.03488* (2020).
[7] Pauline Bennet, Carola Doerr, Antoine Moreau, Jeremy Rapin, Fabien Teytaud, and Olivier Teytaud. 2021. Nevergrad: black-box optimization platform. *ACM SIGEVOlution* 14, 1 (2021), 8–15.
[8] Laurens Bliek, Arthur Guijt, Rickard Karlsson, Sicco Verwer, and Mathijs De Weerdt. 2023. Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. *Applied Soft Computing* 147 (2023), 110744.
[9] Jack Brimberg, Pierre Hansen, N Mladenovic, and Said Salhi. 2008. A survey of solution methods for the continuous location-allocation problem. *International Journal of Operations Research* 5, 1 (2008), 1–12.
[10] Theodore Brown, Alexandru Cioba, and Ilija Bogunovic. 2024. Sample-efficient bayesian optimisation using known invariances. *Advances in Neural Information Processing Systems* 37 (2024), 47931–47965.
[11] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. 1993. On the geometry of feedforward neural network error surfaces. *Neural computation* 5, 6 (1993), 910–927.
[12] Jacob de Nobel, Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. 2021. 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*. 1375–1384.
[13] Jacob de Nobel, Furong Ye, Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. 2024. Iohexperimenter: Benchmarking platform for iterative optimization heuristics. *Evolutionary Computation* 32, 3 (2024), 205–210.
[14] Sofianos Panagiotis Fotias, Ismail Ismail, and Vassilis Gaganis. 2024. Optimization of Well Placement in Carbon Capture and Storage (CCS): Bayesian Optimization Framework under Permutation Invariance. *Applied Sciences* 14, 8 (2024), 3528.
[15] Pasi Fränti and Sami Sieranoja. 2018. K-means properties on six clustering benchmark datasets. , 4743–4759 pages. http://cs.uef.fi/sipu/datasets/
[16] Marcus Gallagher. 2016. Towards improved benchmarking of black-box optimization algorithms using clustering problems. *Soft Computing* 20 (2016), 3835–3849.
[17] Marcus Gallagher. 2019. Fitness landscape analysis in data-driven optimization: An investigation of clustering problems. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2308–2314.
[18] Sara Hajari and Marcus Gallagher. 2024. Searching for Benchmark Problem Instances from Data-Driven Optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 139–142.
[19] Nikolaus Hansen. 2008. Adaptive encoding: How to render search coordinate system invariant. In *International Conference on Parallel Problem Solving from Nature*. Springer, 205–214.
[20] 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.
[21] 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/document
[22] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.
[23] Robert Hecht-Nielsen. 1989. *Neurocomputing*. Addison-Wesley.
[24] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. 1999. Data clustering: a review. *ACM computing surveys (CSUR)* 31, 3 (1999), 264–323.
[25] Maurice George Kendall. 1948. Rank correlation methods. (1948).
[26] Pascal Kerschke, Marcus Gallagher, Mike Preuss, and Olivier Teytaud. 2019. The Machine Learning and Data Analysis (MLDA) Problem Set, v1.0. (2019). Part of GECCO workshop UMLOP.
[27] Fu Xing Long, Bas van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. 2022. Learning the characteristics of engineering optimization problems with applications in automotive crash. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1227–1236.
[28] Manuel López-Ibáñez, Juergen Branke, and Luís Paquete. 2021. Reproducibility in evolutionary computation. *ACM Transactions on Evolutionary Learning and Optimization* 1, 4 (2021), 1–21.
[29] Manuel López-Ibáñez, Diederick Vermetten, Johann Dreo, and Carola Doerr. 2024. Using the empirical attainment function for analyzing single-objective black-box optimization algorithms. *IEEE Transactions on Evolutionary Computation* (2024).
[30] 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*. 829–836.
[31] Kevin P. Murphy. 2022. *Probabilistic Machine Learning: An introduction*. MIT Press. http://probml.github.io/book1
[32] Daiki Otaki, Hirofumi Nonaka, and Noboru Yamada. 2022. Thermal design optimization of electronic circuit board layout with transient heating chips by using Bayesian optimization and thermal network model. *International Journal of Heat and Mass Transfer* 184 (2022), 122263.

[33] Jiming Peng and Yu Wei. 2007. Approximating k-means-type clustering via semidefinite programming. *SIAM journal on optimization* 18, 1 (2007), 186–205.
[34] Michael JD Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal* 7, 2 (1964), 155–162.
[35] Raphael Patrick Prager and Heike Trautmann. 2023. Nullifying the inherent bias of non-invariant exploratory landscape analysis features. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 411–425.
[36] Raphael Patrick Prager and Heike Trautmann. 2024. Pflacco: Feature-based landscape analysis of continuous and constrained optimization problems in Python. *Evolutionary Computation* 32, 3 (2024), 211–216.
[37] Jérémy Rapin, Marcus Gallagher, Pascal Kerschke, Mike Preuss, and Olivier Teytaud. 2019. Exploring the MLDA benchmark on the nevergrad platform. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1888–1896.
[38] Matthew Stephens. 2000. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 62, 4 (2000), 795–809.
[39] Mohammad-H Tayarani-N and Adam Prügel-Bennett. 2013. On the landscape of combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation* 18, 3 (2013), 420–434.
[40] Sébastien Verel, Gabriela Ochoa, and Marco Tomassini. 2010. Local optima networks of NK landscapes with neutrality. *IEEE Transactions on Evolutionary Computation* 15, 6 (2010), 783–797.
[41] Diederick Vermetten, Catalin-Viorel Dinu, and Marcus Gallagher. 2025. Reproducibility and additional files. *Zenodo* (2025). https://doi.org/10.5281/zenodo.15302416
[42] Diederick Vermetten, Jeroen Rook, Oliver L Preuß, Jacob de Nobel, Carola Doerr, Manuel López-Ibañez, Heike Trautmann, and Thomas Bäck. 2025. MO-IOHinspector: Anytime Benchmarking of Multi-Objective Algorithms using IOHprofiler. In *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 242–256.
[43] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods* 17, 3 (2020), 261–272.
[44] Hao Wang, Diederick Vermetten, Furong Ye, Carola Doerr, and Thomas Bäck. 2022. IOHanalyzer: Detailed performance analyses for iterative optimization heuristics. *ACM Transactions on Evolutionary Learning and Optimization* 2, 1 (2022), 1–29.
[45] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)* 23, 4 (1997), 550–560.