Towards Constructing a Suite of Multi-objective Optimization Problems with Diverse Landscapes

Andrejaana Andova[1,2(✉)], Tobias Benecke[3], Harald Ludwig[4], and Tea Tušar[1,2]

[1] Jožef Stefan Institute, Ljubljana, Slovenia
{andreajaana.andova,tea.tusar}@ijs.si
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[3] Otto-von-Guericke-University Magdeburg, Magdeburg, Germany
tobias.benecke@ovgu.de
[4] Johannes Kepler University Linz, Linz, Austria
harald.ludwig@jku.at

Abstract. Given that real-world multi-objective optimization problems are generally constructed by combining individual functions to be optimized, it seems sensible that benchmark functions would also follow this procedure. Since the pool of functions to choose from is large and the number of function combinations increases exponentially with the number of objectives, we need a smart way to choose a reasonably sized and diverse collection of function combinations to use in benchmarking experiments. We propose a four-step approach that analyzes the landscape characteristics of all function combinations and selects only the most diverse ones to form a suite of problems. In this initial study, we test this idea on the pool of `bbob` functions and the case of two objectives. We provide a proof of concept for the proposed approach and its initial results. We also discuss its limitations to be addressed in future work.

Keywords: benchmarking · test problems · multi-objective optimization · exploratory landscape analysis

1 Introduction

One of the purposes of benchmarking is to gain knowledge about algorithm performance on various test problems, which can be applied when solving real-world optimization problems. The latter are often computationally expensive, making it intractable (if not impossible) to perform extensive experiments to find the best algorithm for the given problem. The more realistic scenario in such cases is to find some information about the properties of the real-world problem at hand and solve it using the algorithm that performs best on quick-to-evaluate test problems with similar characteristics.

A. Andova, T. Benecke and H. Ludwig—Contributed equally to this work.

J. Correia et al. (Eds.): EvoApplications 2023, LNCS 13989, pp. 442–457, 2023.
https://doi.org/10.1007/978-3-031-30229-9_29

A good choice of benchmark problems is crucial to make this scenario useful in practice. According to [1], a suite of benchmark problems should be *diverse*, containing problems with different characteristics, and *representative* by including difficulties that are found in real-world problems. Its problems should be *scalable and tunable* and, in order to facilitate performance evaluation, have *known optima* or at least some *known best performance*. Ideally, the suite would be *continuously updated* to prevent the over-fitting of algorithms.

In continuous single-objective optimization, the well-known `bbob` test suite [6] satisfies all these requirements. However, there is no equivalent of such a suite in multi-objective optimization (the `bbob-biobj` suite [3] is limited to bi-objective problems). The decades old, still most often used benchmark suites in multi-objective optimization, namely ZDT [16], DTLZ [5] and WFG [10], are built following the *bottom-up approach*, meaning they are designed around the desired properties of the Pareto-front, which sacrifices real-world relevance of resulting problems to a simple suite construction procedure and controllable problem properties [4]. Note that because of the prevalence of these suites in multi-objective optimization, algorithms have been overfitting to their problems [11], further motivating the need for a change.

We believe that the alternative approach to problem construction, which uses single-objective test function combinations to create multi-objective problems, more closely resembles the real-world conditions and can be used to create a diverse benchmark suite without the biases brought on by the bottom-up approach. However, simply creating all function combinations from a pool of functions is infeasible due to the high number of resulting problems. With k functions to choose from, we can construct $\binom{k+m-1}{m}$ unique multi-objective function combinations (without including permutations of the same functions), where m is the number of objectives. For example, given $k = 10$ functions, we get 55 function combinations with two objectives, 2002 function combinations with five objectives and 92 378 function combinations with ten objectives. This is not sustainable and a more sophisticated strategy is needed to select good function combinations to form a reasonably sized suite of multi-objective benchmark problems.

In this paper, we propose to use problem landscape characteristics to guide to the selection of a manageable number of function combinations. The idea is to compute problem (dis)similarity using exploratory landscape analysis (ELA). This meets the first requirement for a suite of benchmark problems—its diversity. To make sure that function combinations are also representative of real-world difficulties, we use a pool of functions with this quality, the `bbob` functions (which are also scalable, tunable, and have known optima). We try this idea in the case of two objectives. Without a formal way to validate such a construction, we look at the resulting problems as well as some intermediate steps in the construction procedure from multiple perspectives to understand the implications of our choices and find ideas for future improvements.

In the following pages, Sect. 2 provides some background on multi-objective optimization problems, the `bbob` and `bbob-biobj(-ext)` benchmark suites and the exploratory landscape analysis features used to characterize the problems. Section 3 details the employed four stepped methodology: (i) generating the base

set of problems, (ii) computing their ELA features, (iii) selecting a diverse subset of problems, and (iv) generating similar instances. The results of this procedure are presented in Sect. 4, where we first validate our idea on 2-D problems, for which the problem landscape can be visualized, then show the results for all considered dimensions and finally discuss the limitations of our approach and ideas to improve it. Section 5 concludes the paper with final remarks.

2 Background

In this section, we first clarify the terminology around multi-objective optimization problems, followed by a brief introduction to the single-objective `bbob` functions used to construct the bi-objective problems in this paper. Finally, the ELA features used to characterize the fitness landscapes are shortly presented.

2.1 Multi-objective Optimization Problems

We are concerned with multi-objective optimization problems of the form:

$$\text{minimize} \quad F^{\theta}(x) = (f_1^{\theta_1}(x), \ldots, f_m^{\theta_m}(x)), \tag{1}$$

where $x = (x_1, \ldots, x_n) \in S$ is a search vector from the n-dimensional search space $S \subseteq \mathbb{R}^n$ and $f_i^{\theta_i} : S \to \mathbb{R}$, $i = 1, \ldots, m$, are parameterized objective functions, where $m > 1$ and $\theta = (\theta_1, \ldots, \theta_m) \in \Theta$ parameterizes the function instance.

We use the term *function combination* to denote the non-instantiated m-tuple of functions $F(x) = (f_1(x), \ldots, f_m(x))$, while a *problem* is a particular instance of the function combination, $F^{\theta}(x) = (f_1^{\theta_1}(x), \ldots, f_m^{\theta_m}(x))$. Different instances of a function might be shifted in the decision and/or objective space, can be rotated, etc. and are used to test algorithm (in)variability to these changes.

A problem solution $x \in S$ *dominates* another solution $y \in S$ when $f_i^{\theta_i}(x) \leq f_i^{\theta_i}(y)$ for all $i = 1, \ldots, m$ and $f_j^{\theta_j}(x) < f_j^{\theta_j}(y)$ for at least one $j = 1, \ldots, m$. A solution $x^* \in S$ is *Pareto optimal* if there are no solutions $x \in S$ that dominate x^*. All non-dominated solutions represent the *Pareto set* P of the problem, while its image in the objective space is called the *Pareto front*. Additionally, the *ideal point* z^{ideal} in the objective space $\mathbb{R}^m$ is defined as the point whose coordinates equal the optimal values of $f_i^{\theta}(x)$ for each $i = 1, \ldots, m$ independently. That is, $z^{\text{ideal}} = (\inf_{x \in S} f_1^{\theta_1}(x), \ldots, \inf_{x \in S} f_m^{\theta_m}(x))$. Conversely, the *nadir point* z^{nadir} in the objective space $\mathbb{R}^m$ consists in each objective of the worst value obtained by a Pareto optimal solution. That is, $z^{\text{nadir}} = (\sup_{x \in P} f_1^{\theta_1}(x), \ldots, \sup_{x \in P} f_m^{\theta_m}(x))$.

2.2 The `bbob` Functions

The `bbob` function suite contains 24 well-known and understood functions in six dimensions (2, 3, 5, 10, 20, and 40) and 15 instances that change through the years [6]. The functions have known optima and incorporate various difficulties found in real-world problems. They were carefully selected to support a variety of research questions and are categorized into five groups based on their properties:

– separable functions (functions f_1 to f_5),
– functions with low or moderate conditioning (f_6 to f_9),
– highly conditioned and unimodal functions (f_{10} to f_{14}),
– multimodal functions with global structure (f_{15} to f_{19}), and
– multimodal functions with weak global structure (f_{20} to f_{24}).

The `bbob` function suite is implemented in the Comparing Continuous Optimizers (COCO) platform [9], which supports automated benchmarking using these problems. Instances are used to measure robustness of stochastic and deterministic optimizers alike. So in one benchmarking experiment, an algorithm is run once on a total of $24 \cdot 6 \cdot 15 = 2160$ problems.

Two suites of bi-objective problems formed as combinations of the `bbob` functions were proposed in [3]. The first one, `bbob-biobj`, contains all function combinations of ten manually chosen `bbob` functions (two from each group), resulting in 55 function combinations. The second one, `bbob-biobj-ext`, extends this selection by adding additional function combinations where both functions are different and come from the same group, resulting in 92 function combinations in total. This extension was proposed to increase the diversity of the problems and therefore uses the pool of all but one of the 24 `bbob` functions (it leaves out the Weierstrass function f_{16} because it has multiple global optima, making the computation of the nadir point intractable). So, in a bi-objective benchmarking experiment, an algorithm needs to optimize $55 \cdot 6 \cdot 15 = 4950$ problems of the `bbob-biobj` suite or $92 \cdot 6 \cdot 15 = 8280$ problems of the `bbob-biobj-ext` suite. Both numbers are high and would increase drastically if the same procedure was used to form suites of problems with more objectives.

When `bbob` functions are combined to form multi-objective problems, two new issues appear [3]. The first is that we no longer know all optimal solutions. The single-objective optima of the functions reveal only the extreme points of the Pareto front, not the entire Pareto front (nor set). This makes performance assessment more challenging. The second issue (addressed in Sect. 3.4) is that the instances of function combinations differ among themselves more than in the case of single-objective `bbob` problems, especially when one or both of the functions are multimodal. This defies the purpose of instances as 'minor' modifications that should not significantly affect the performance of robust algorithms [8].

2.3 ELA Features

Exploratory landscape analysis is a method that extracts information from the problem landscape [15]. The information is gathered from the objective values of a sample of problem solutions and represented by so-called *features* (numerical values) that characterize the landscape. Hundreds of ELA features can be computed for continuous single-objective problems, see for example the `flacco` package [12], while the only collection of ELA features for continuous multi-objective problems is (to the best of our knowledge) the one presented in [13]. Its implementation is available from https://gitlab.com/aliefooghe/landscape-features-mo-icops/.

Although the features from [13] were originally proposed for multi-objective interpolated continuous optimization problems, they can also be applied to 'standard' multi-objective optimization problems and are consequently used in this paper. The features are categorized into four types: *global*, *multimodality*, *evolvability*, and *ruggedness* features.

The *global landscape features* extract information about the global properties of the multi-objective landscape. Among the calculated properties we find the correlation between two objectives, the average and maximum distance among sampled solutions in the search and the objective space, the proportion of non-dominated solutions, the hypervolume value, etc.

The *features that characterize the landscape multimodality* measure the properties of the local optima. Two different types of local optima are considered: single-objective local optima and multi-objective local optima. The neighborhood of a solution (the set of the closest samples to this solution) is used to detect the single-objective local optima—if no neighbor of the target solution has a better objective value than it, then the target solution is a local optimum. Similarly, a multi-objective local optimum dominates all its neighboring solutions. The multimodality features measure the percentage of solutions that are local optima, their distance, etc.

The *features that characterize the landscape evolvability* measure the proportion of neighbors of a solution that can outperform it. First, the ELA method records for each solution the proportion of neighbors that dominate, are dominated by, or are mutually non-dominated with the corresponding solution. Then, it applies some statistical measures to determine how these numbers differ in the entire sample of solutions.

The last type of features are the *ruggedness features*, which measure the ruggedness of the problem landscape by analyzing the correlation of the evolvability features among neighboring solutions.

3 Methodology

We wish to follow the benchmarking procedure used by the COCO platform [8], which assumes that the function combinations contained in a benchmarking suite are instantiated in several dimensions and instances. The dimensions are needed to test the scalability of the algorithms, while the instances are used to assess their repeatability. This means that we need to compare the properties of problems for multiple dimensions at the same time, as well as produce instances that are very similar to each other (which requires effort, as by default, the multi-objective problem instances can be quite different).

Our procedure to construct a diverse suite of bi-objective problems therefore consists of the following four steps, described in more detail in the rest of this section:

1. Generate 15 instances of all function combinations in all considered dimensions (our base set of problems),
2. Compute the ELA features for all these problems,

3. Select a small suite with diverse problems, and
4. Generate additional similar instances for the selected problems.

3.1 Generating the Base Set of Problems

To generate our base bi-objective problem set, we start with all possible combinations of the `bbob` functions [6]. However, similarly as in the `bbob-biobj-(ext)` test suites [3], we remove the Weierstrass function because it has multiple global optima, complicating the computation of the nadir point. Furthermore, we regard both permutations of two functions to be the same, as they produce the same landscapes. This results in 276 unique function combinations.

Instances of the problems are also generated using the same approach as in the `bbob-biobj-(ext)` test suites [3]. This means that we only consider a bi-objective instance if the single-objective optima have at least the distance of 10^{-4} in the search space, and if the ideal point and the nadir point have at least the distance of 10^{-1} in the non-normalized objective space. These two conditions need to be satisfied for all considered dimensions (2, 3, 5, 10, and 20). We generate 15 different instances for each problem. Considering 276 unique problems with 15 instances in 5 different dimensions results in in 20 700 problems overall.

3.2 Computing the ELA Features

To generate the ELA features described in Sect. 2.3, a sample of solutions is needed. We create it using the Latin Hypercube Sampling method [14], which is one of the most widely used sampling methods in evolutionary computation. To assure stability (avoid that the differences among problem features originate from different samples rather than the different problem landscape properties), we use the same sample for all problems of the same dimension. The sample size is $200 \cdot n$, where n is the search space dimension. We evaluate the samples for each of the five dimensions considered (2, 3, 5, 10, 20) on all problems.

In the next step, we normalize the objective values for each problem using its ideal and nadir points. The ideal point represents the minimal and the nadir point the maximal objective values for normalization. We thus use the min-max normalization on the objective values for each of the solutions in the sample. In this way, the objective values of all problems are comparable. Note however that since the nadir point is used for the normalization maximum, solutions worse than the nadir will have objective values larger than 1.

We then compute 48 features using the code mentioned in Sect. 2.3 with default parameters. This means that the reference point for hypervolume computation equals $(1, 1)$ and only one neighbor is used in neighborhood-based features. For each of the $276 \cdot 15 = 4140$ problem combinations we concatenate the features of all five dimensions to a single feature vector of size 240.

Based on the computed features, we finally filter out some of the unusable features and/or problem instances. The filtering is triggered by one of the following reasons: Firstly, if the feature values are the same for all problems. This happens because some of the features are computed based on the variance of

the samples in the search space. Since we use the same sample for all problems of the same dimension, such features are useless in our case and thus removed. Secondly, if the features values cannot be computed because the sample has a single non-dominated solution. Some of the features require multiple non-dominated solutions to compute. Although we are creating problem instances where the Pareto front is not 'too small', this still happens occasionally. As we consider such problems not interesting from the perspective of multi-objective optimization (they are not desirable in the test suite), we remove them from the collection. Lastly, if the feature values cannot be computed because we use only one neighbor (the default setting). Such features are also removed. In this way, we get the feature vector of size 202 and 4030 problem instances.

3.3 Selecting a Diverse Subset of Problems

The general idea of selecting a diverse subset of problems used in this paper is to do so iteratively, always choosing the problem that is most different in terms of ELA features to the ones already selected. Therefore, we divide the set of all problems into the subsets of *selected* and *unselected* problems. Recall that the 48 ELA features for each dimension are combined into a single, large feature vector so that the results of all dimensions are included in this selection process.

Because of the high dimensionality of the feature vector, we compare the problems using cosine similarity instead of some distance metric. Distance metrics would demonstrate stark differences because slight variations in individual dimensions can accumulate over dimensions. When comparing two vectors, cosine similarity only considers their angle rather than their magnitude, making it immune to this issue. To determine the similarity between two bi-objective problem instances, we compute a similarity matrix where the cosine similarity ranges from 0 (completely unrelated) to 1 (identical problems).

The first problem in this iterative process needs to be selected by hand. In our work, we chose the first instance of the double sphere problem as the starting point. The double sphere problem is one of the easiest bi-objective problems to solve and the only one for which we know the entire Pareto set (the line segment between the two single-objective optima) and front (the image of this line segment in the objective space). We believe all suites should contain a well-understood problem like this, so this is a natural starting point for our procedure.

The remaining problems are chosen from the unselected subset as those that are the most different from the already selected. This is achieved as follows. First, we find for each unselected problem the most similar selected one and record their similarity. Then, we compare these similarities and choose the unselected problem with the lowest recorded similarity. In other terms, we try to maximize the minimum similarity between the already selected problems and the ones to be chosen next. This way, we iteratively add the most dissimilar problem to our selected subset.

This approach can generate an arbitrary large subset of benchmark problems. We need to decide when to stop this procedure, i.e., how many function combinations we wish to have in the suite. This is nontrivial and somewhat

preference-based, as it requires choosing a trade-off between the suite diversity and representation abilities on the one hand, and the computational cost and evaluation time on the other. If the desired suite size is known, the proposed greedy procedure could also be replaced by a more sophisticated algorithm.

3.4 Generating Similar Instances

Different instances of the same function combination have varying fitness landscapes (and consequently features). This can be problematic when creating a benchmark suite where the instances are supposed to test algorithm repeatability. To solve this problem, we try to find 14 instances that are very similar to the selected problem instance. This way, we can expect similar algorithm performance on all instances of the selected problems.

We generate the problem instances for the N most diverse problems selected (let us call them *target problems*) in a post-processing fashion. For each target problem, we first generate 140 additional instances, the same as in Sect. 3.1. Then, we compute the ELA features as described in Sect. 3.2 on these new instances. This gives us 155 instances for each problem, categorized by the ELA features. Once again, we use cosine similarity to measure the similarity between the target problem and each additional instance. Finally, we select 14 instances that are most similar to the target problem, yielding 15 similar instances.

4 Results and Discussion

In this section, we present the results of applying our methodology on the pool of the `bbob` functions. We first show the results of a test run on only the 2-D problems as a proof of concept for our approach. Then we present the actual results on multiple dimensions. Finally, we list the limitations of our approach, proposing ideas to improve it in the future.

4.1 Proof of Concept on 2-D Problems

Given that there is no established way to validate our proposed approach, we try to gain a better understanding of its workings by applying it only to 2-D problems, whose landscapes can be visualized. This means that we perform the entire procedure described in Sect. 3, with the exception of the parts that require stacking together the features from problems of different dimensions.

We first select an arbitrary function combination that is also included in the `bbob-biobj` test suite and therefore has some known properties (see [3] and its supplementary material website at https://numbbo.github.io/bbob-biobj/ for more information). This is the function combination (f_1, f_8), where f_1 is the sphere function and f_8 is the original Rosenbrock function, instantiated with 15 different instances. We use it to visually verify that the way we compute problem features and measure their similarity works as intended.

Figure 1 presents the problem similarity heatmap for the 15 instances of (f_1, f_8) where the rows and columns are sorted in such a way that the similar

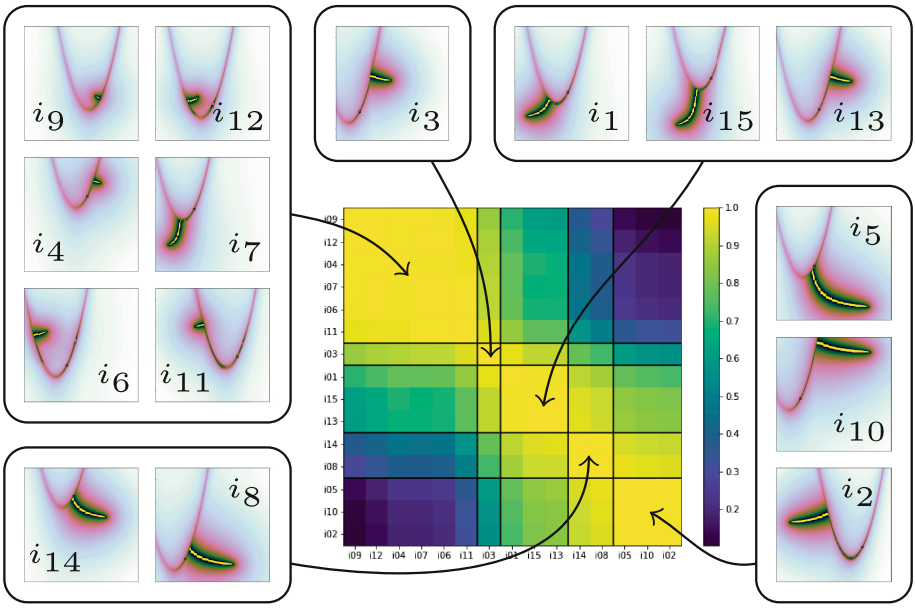

Fig. 1. Problem similarity heatmap for the 15 instances of the 2-D function combination (f_1, f_8) (function F_4 in the `bbob-biobj` suite) and their corresponding landscape visualizations using dominance ranking ratio [3,7] show that the size of the Pareto set (yellow curves) is proportional to the similarity among instance. The instances are numbered as in the `bbob-biobj` suite. (Color figure online)

instances (yellow hues) are placed together. Furthermore, the problem landscape of the different instances is also visualized using dominance rank ratio plots [3,7]. In these plots, the search space is approximated by the grid points and their color denotes the proportion of other grid points that dominate the current one (the more such points, the lighter the color). Additionally, yellow is used to denote the Pareto set approximation (non-dominated grid points).

First, we notice that the problem instances could be grouped into five clusters based on the values of the similarity matrix. Next, we see that the size of the Pareto set is proportional to the similarity among instances. On the one hand we have six mutually similar instances i_9, i_{12}, i_4, i_7, i_6, and i_{11}, with short Pareto sets (top left), and on the other hand, three mutually similar instances i_5, i_{10}, and i_2 with long Pareto sets (bottom right). The instances in the first group are very different from the instances in the last one according to the similarity matrix as well as their landscape visualizations.

The characterization of the `bbob-biob-(ext)` problems in [3], where the first five instances of all 92 problems were visually inspected for five properties (number of Pareto set subsets, number of Pareto front subsets, convex Pareto front, Pareto set outside of $[-5, 5]^2$ and number of basins of attraction),

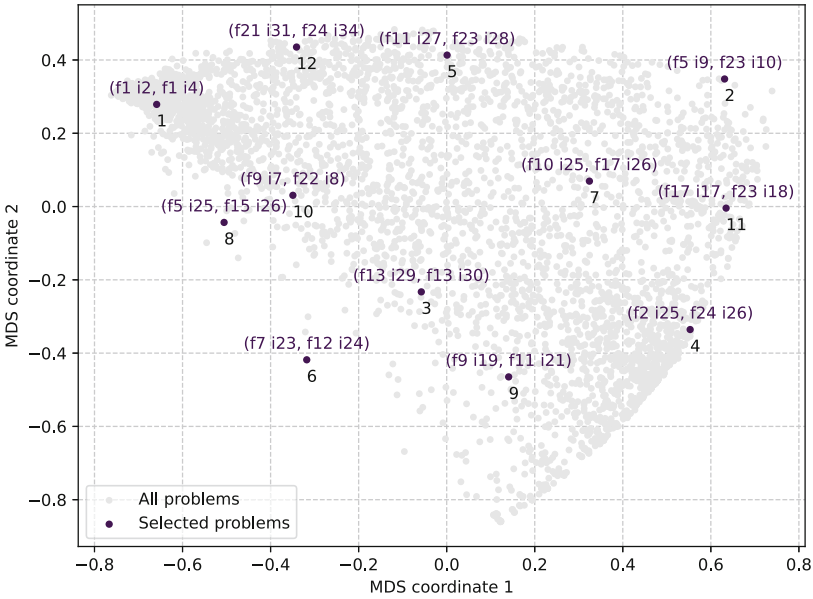

Fig. 2. Multidimensional scaling of the 48-D space of features to a plane for all 2-D problems. The 12 most diverse problems (see Fig. 3) are emphasized and labeled with their consecutive number in black and function instance combination in purple. (Color figure online)

has found no difference between the first five instances of the 2-D function combination (f_1, f_8). In this sense, our approach exhibits higher discriminability powers as it does differentiate between these instances, in fact, they reside in four of the five denoted clusters. We therefore see that the applied ELA features and the cosine similarity metric properly characterize the instances of this function combination, showing promise of this methodology.

If we use the proposed approach to construct a suite of 12 most diverse 2-D function combinations, we get the results shown in Fig. 2. This plot presents a Multidimensional Scaling (MDS) projection of all problems from the 48-D space of ELA features to a plane. MDS chooses a projection so that the distances in the projected space respect the distances well in the original space (we used the default MDS implementation from the `sklearn` Python library to produce this plot). The darker dots denote our selected function instance combinations. We can see that they are relatively uniformly distributed over the entire space, although any such projection needs to be interpreted with caution.

One interesting thing to note is that some `bbob` functions are included in multiple combinations, see for example f_{23}, which was employed three separate times. This is somewhat counter-intuitive, as we could expect the functions to repeat themselves only after (almost) all different available functions have been selected. This might be caused by large differences in some function instances.

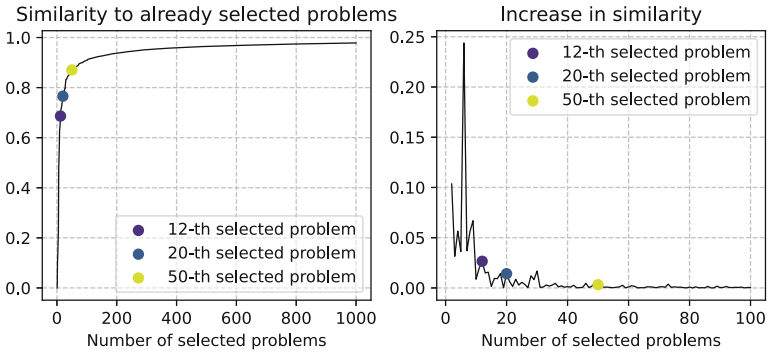

Fig. 3. Similarity to already selected problems when adding new problems to the selection. The plot on the left shows the similarity itself (which has a large increase at first and then flattens around 100 problems), while the plot on the right shows the increase in similarity compared to the previous value. In both plots, the 12th, 20th and 50th problems are emphasized.

4.2 Results on Multiple Dimensions

Deciding on the number of problems to include in a benchmark requires compromising. On the one hand, we want to select many problems which cover a wide range of different characteristics. On the other hand, the benchmark should be of a manageable size to be fit for use in large benchmarking studies. As we can vary the number of selected problems in our approach, it is beneficial to evaluate the impact or usefulness of adding each problem by calculating its similarity to the already selected ones.

Figure 3 shows this in two plots. On the left-hand side plot, we can observe the similarity of a newly selected problem to the already selected ones for the first 1000 problems. The right-hand side plot shows the increase in similarity for each newly selected problem for the first 100 problems. On the left-hand side plot, a logarithmic growth can be observed, increasing rapidly for the first 100 problems selected, then gradually flattening out. On the plot on the right, we can see a rugged decline in the similarity increase for all problems, with a notable peak at the sixth selected problem. Not surprisingly, when only a small number of problems are included in the suite, it is very beneficial to add more. Selecting less than 12 problems, therefore, seems undesirable. We see that the similarity to the already selected problems increases more slowly after the first 12 problems have been selected. However, the data does not provide a clear cut-off point after which selecting more problems becomes less effective.

The number of problems selected cannot be determined clearly as it depends on many factors. In this paper, 12 problems were selected to strike a balance between diversity, representation, and and the ability to execute and show results of benchmarking experiments in the scope of scientific work.

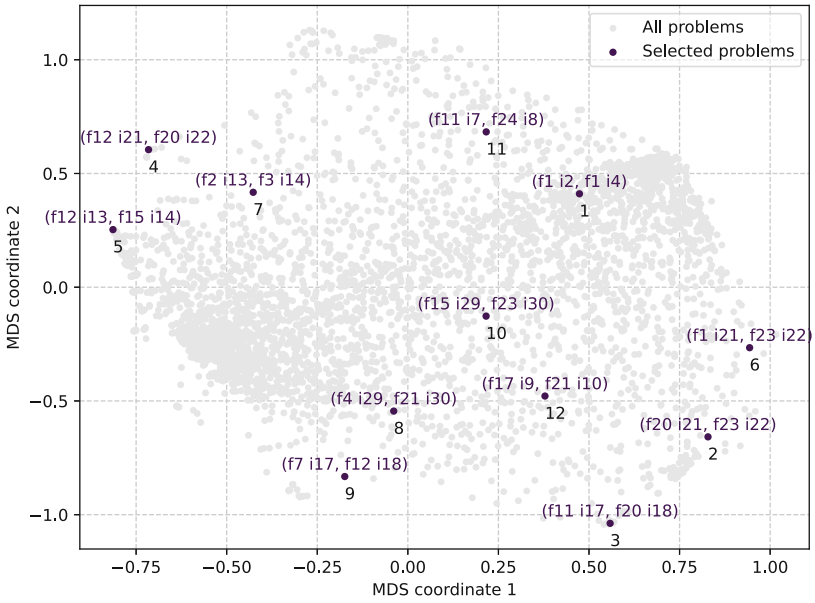

Fig. 4. Multidimensional scaling of the 202-D space of features to a plane for problems with multiple dimensions. The 12 most diverse problems are emphasized and labeled with their consecutive number in black and function instance combination in purple. (Color figure online)

Similar to the results on the 2-D problems in Fig. 2, Fig. 4 shows the multidimensional scaling of the ELA features to a plane for the problems in all used dimensions. We can again notice that the 12 most diverse problems resulting from our approach are rather uniformly distributed over the space, although given that the feature space is 202-D in this case, care must be taken when making assumptions based on such visualizations.

From the point of view of included `bbob` functions, we can see that even more (compared to the previous section) have been selected multiple times. That is, each of f_{11}, f_{12}, f_{20} and f_{23} has been used three times in the top 12 most diverse problems. We are currently unable to explain why some repeated functions are preferred over others that have not yet been included.

Finally, Fig. 5 visualizes with violin plots the distribution of the cosine similarity values between the target problem (one of the 12 problem instances selected to be included in the suite) and all 155 instances of the same problem (in blue) as well as between the target problem and the closest 15 instances of the same problem (in orange). This is done for all 12 target problems. Note that the 155 problem instances are the result of the procedure described in Sect. 3.4.

We can observe that the similarity of the closest instances differs from one function combination to the next. On some combinations, e.g., (f_{12}, f_{20}) and (f_{12}, f_{15}), they are very close, while for others (notably the double sphere function (f_1, f_1), where one of the closest instances is rather far away from the target

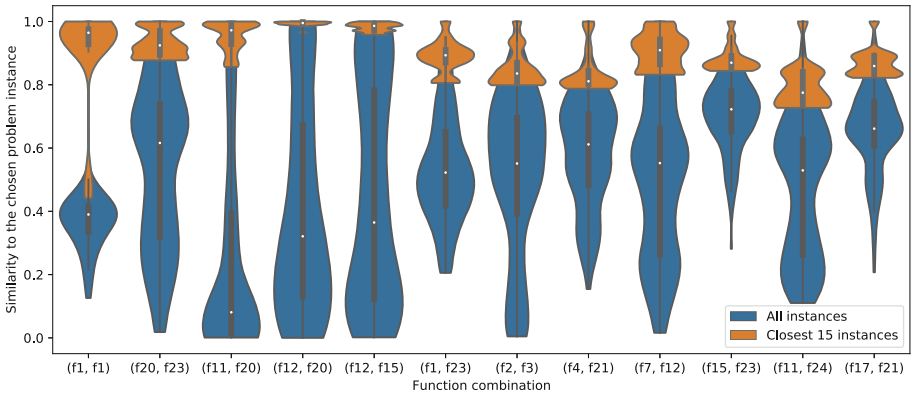

Fig. 5. Distribution of cosine similarity values between each target problem and all 155 corresponding problem instances (in blue), and the target problem and its closest 15 instances (in orange). (Color figure online)

problem) this is not the case. This means that further instances would need to be generated on such problems to achieve better similarity of the top 15 instances.

Additional insights into the fitness landscapes can be gained by analyzing the distributions in the blue violin plots. For example, in the function combination (f_1, f_1) we see two groups forming, one close to the target problem in the similarity range of $[0.8, 1.0]$ and one more distant with a median similarity at around 0.4. This indicates that different instances of this function combination might result in two different problem groups in terms of the ELA features. Interestingly, (f_1, f_{23}), which likewise includes the sphere function f_1, also seems to produce two clusters of problems. On the other hand, some function combinations seem to produce very diverse instances that cover the entire range of similarity to the target problem, like (f_{12}, f_{20}) and (f_{12}, f_{15}). Other combinations, like (f_{15}, f_{23}) and (f_{17}, f_{21}) show only one large group with some outlying instances. A more elaborate evaluation is needed to understand why this happens.

This analysis has shown that a fixed number of function instances is, in general, not enough to achieve close instances. An iterated approach where instances are generated until a closeness threshold has been met might work better (although it could also take a long time).

4.3 Limitations

Our work presents a first attempt at creating a new benchmarking suite by using ELA features to ensure its diversity. While we were able to show that the idea works quite well (especially on the 2-D problems), we acknowledge some limitations of this initial study that need to be addressed in the future.

Our procedure *assumes that all problems are instantiated and characterized* before the selection takes place. Currently, calculating the ELA features is the bottleneck of the proposed methodology. If many more function combinations

would have to be explored (for example, in the case of many-objective problems), this could become intractable and an alternative approach that does not require characterizing all function combinations would need to be implemented (perhaps by filtering unpromising function combinations based on the properties of single functions or combinations of two functions).

No quantifiable goal and/or evaluation of the proposed approach exists. Several concepts that we use throughout the study, such as problem diversity, instance closeness, and suite size, are hard to quantify in terms of thresholds that denote a satisfactory value. This means that it is also hard to judge whether our procedure was able to meet its goals. One particular issue that we need to explore is whether the methodology optimizes for outliers. This could result in a suite that is not representative of real-world conditions and therefore go against our goal. One way to evaluate the usefulness of a benchmark suite is to show that it differentiates between algorithms. We will address this in future work.

Sensitivity to Parameters. While we have not tested this extensively yet, our experiments suggest that the resulting problem selection strongly depends on various parameters of our approach, such as the selection of the applied ELA features, the choice of dimensions to be included in the construction, the number and variety of instances, the initial selected problem, etc. This sensitivity would first need to be studied more comprehensively and then decreased where possible (although, of course, the reliance on some parameters cannot be eliminated).

Questionable Scalability in Search Space Dimension. With increasing search space dimension, the sample size used for computing the ELA features increases only linearly, which seems not to be enough for properly categorizing high-dimensional problems. We need to look into this issue more deeply.

Questionable Scalability in Objective Space Dimension. So far, the approach has only been tested on bi-objective problems. In order for the resulting benchmark to be truly representative of the real world, where many problems have more than two objectives [2], the approach should be tested also using three, five and more objectives. This is closely tied with the first limitation.

Using Default ELA Features. While using default ELA features was the 'safe choice' for this set of initial experiments, we have realized that it might have been flawed. For example, a lot of the used features might not contain useful information for predicting algorithm performance, and thus, applying feature selection methods would remove the unnecessary noise in the feature space. Furthermore, we currently used the default setting for calculating the ELA features. However, now the hypervolume reference point coincides with the nadir point, meaning that all non-dominated solutions that are worse than the nadir do not contribute to the hypervolume. We should therefore be looking at alternative hypervolume measures, such as the indicator $I_{\mathrm{HV}+}$ from [8], which resolves this issue. Moreover, using only one neighbor might be questionable, we need to explore whether having two or three neighbors works better.

5 Conclusions

Designing multi-objective benchmark problems as combinations of individual single-objective functions closely follows the construction of real-world problems and should therefore be preferred to the bottom-up approach. However, one cannot simply employ all possible function combinations, as they are too many to be usable in practice and a smart way to choose a reasonably sized and diverse collection of function combinations is needed.

In this work, we proposed to use problem landscape characteristics (computed as ELA features) to create a benchmark suite of diverse and representative multi-objective optimization problems of the chosen size. The main idea is to construct the suite by adding problems whose cosine similarity in the ELA feature space to the already selected problems is minimal. The approach was tried on bi-objective combinations of the `bbob` functions, which are scalable, tunable, and contain difficulties found in real-world problems.

We first used a simplified procedure formulation to prove on 2-D problems that our concept is promising, and then showed results of the actual procedure that uses all considered dimensions. Finally, we listed the limitations of this initial approach that need to be addressed in future work.

Acknowledgements. Special thanks to the Species Society and the organizers of the Species Summer School 2022 for connecting us and making this research possible.

We acknowledge financial support from the Slovenian Research Agency (research project "Constrained multi-objective Optimization Based on Problem Landscape Analysis", young researcher program and research core funding no. P2-0209). This work is also part of the Research Initiative "SmartProSys: Intelligent Process Systems for the Sustainable Production of Chemicals" funded by the Ministry for Science, Energy, Climate Protection and the Environment of the State of Saxony-Anhalt.

References

1. Bartz-Beielstein, T., et al.: Benchmarking in optimization: best practice and open issues. CoRR abs/2007.03488 (2020). https://arxiv.org/abs/2007.03488
2. van der Blom, K., et al.: Towards realistic optimization benchmarks: a questionnaire on the properties of real-world problems. In: GECCO 2020: Genetic and Evolutionary Computation Conference, Companion Volume, pp. 293–294. ACM (2020). https://doi.org/10.1145/3377929.3389974
3. Brockhoff, D., Auger, A., Hansen, N., Tušar, T.: Using well-understood single-objective functions in multiobjective black-box optimization test suites. Evol. Comput. **30**(2), 165–193 (2022). https://doi.org/10.1162/evco_a_00298
4. Brockhoff, D., Tušar, T.: GECCO 2022 tutorial on benchmarking multiobjective optimizers 2.0. In: GECCO 2022: Genetic and Evolutionary Computation Conference, Companion Volume, pp. 1269–1309. ACM (2022). https://doi.org/10.1145/3520304.3533635
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, pp. 825–830. IEEE (2002). https://doi.org/10.1109/CEC.2002.1007032

6. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2010: presentation of the noiseless functions. Tech. Rep. 2009/20, Research Center PPE (2009). http://coco.gforge.inria.fr/downloads/download16.00/bbobdocfunctions.pdf
7. Fonseca, C.M.: Multiobjective genetic algorithms with application to control engineering problems, Ph. D. thesis, University of Sheffield (1995)
8. Hansen, N., Auger, A., Brockhoff, D., Tušar, T.: Anytime performance assessment in blackbox optimization benchmarking. IEEE Trans. Evol. Comput. **26**(6), 1293–1305 (2022). https://doi.org/10.1109/TEVC.2022.3210897
9. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. Optim. Meth. Softw. **36**(1), 114–144 (2021). https://doi.org/10.1080/10556788.2020.1808977
10. Huband, S., Barone, L., While, L., Hingston, P.: A scalable multi-objective test problem toolkit. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 280–295. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31880-4_20
11. Ishibuchi, H., Setoguchi, Y., Masuda, H., Nojima, Y.: Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. IEEE Trans. Evol. Comput. **21**(2), 169–190 (2017). https://doi.org/10.1109/TEVC.2016.2587749
12. Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-Package Flacco. In: Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., Vichi, M. (eds.) Applications in Statistical Computing. SCDAKO, pp. 93–123. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25147-5_7
13. Liefooghe, A., Vérel, S., Lacroix, B., Zavoianu, A., McCall, J.A.W.: Landscape features and automated algorithm selection for multi-objective interpolated continuous optimisation problems. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 421–429. ACM (2021). https://doi.org/10.1145/3449639.3459353
14. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **42**(1), 55–61 (2000)
15. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: 13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 829–836. ACM (2011). https://doi.org/10.1145/2001576.2001690
16. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evol. Comput. **8**(2), 173–195 (2000). https://doi.org/10.1162/106365600568202