Towards Exploratory Landscape Analysis for Large-scale Optimization: A Dimensionality Reduction Framework

Ryoji Tanabe
Yokohama National University & Riken AIP
Yokohama, Kanagawa, Japan
rt.ryoji.tanabe@gmail.com

ABSTRACT
Although exploratory landscape analysis (ELA) has shown its effectiveness in various applications, most previous studies focused only on low- and moderate-dimensional problems. Thus, little is known about the scalability of the ELA approach for large-scale optimization. In this context, first, this paper analyzes the computational cost of features in the flacco package. Our results reveal that two important feature classes (ela_level and ela_meta) cannot be applied to large-scale optimization due to their high computational cost. To improve the scalability of the ELA approach, this paper proposes a dimensionality reduction framework that computes features in a reduced lower-dimensional space than the original solution space. We demonstrate that the proposed framework can drastically reduce the computation time of ela_level and ela_meta for large dimensions. In addition, the proposed framework can make the cell-mapping feature classes scalable for large-scale optimization. Our results also show that features computed by the proposed framework are beneficial for predicting the high-level properties of the 24 large-scale BBOB functions.

CCS CONCEPTS
• Mathematics of computing → Evolutionary algorithms.

KEYWORDS
Exploratory landscape analysis, fitness landscape analysis, large-scale black-box optimization, dimensionality reduction

ACM Reference Format:
Ryoji Tanabe. 2021. Towards Exploratory Landscape Analysis for Large-scale Optimization: A Dimensionality Reduction Framework. In 2021 Genetic and Evolutionary Computation Conference (GECCO ’21), July 10–14, 2021, Lille, France. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3449639.3459300

1 INTRODUCTION

General context. We consider a noiseless black-box optimization of an objective function \( f : \mathbb{X} \rightarrow \mathbb{R} \), where \( \mathbb{X} \subseteq \mathbb{R}^n \) is the \( n \)-dimensional solution space. This problem involves finding a solution \( x \in \mathbb{X} \) with an objective value \( f(x) \) as small as possible without any explicit knowledge of \( f \). Fitness landscape analysis [24, 36] is generally used to understand the high-level properties of a problem.

Table 1: Dimension \( n \) of the BBOB functions in selected previous studies for the following four tasks: high-level property classification (HP), algorithm selection (AS), performance prediction (PP), and per-instance algorithm configuration (PIAC).

| Ref. | Year | Task | Dimension \( n \) |
|------|------|------|-------------------|
| [25] | 2011 | HP   | \( n \in \{5, 10, 20\} \) |
| [32] | 2012 | PP   | \( n \in \{2, 3, 5, 10, 20\} \) |
| [5]  | 2012 | AS   | \( n \in \{5, 10, 20\} \) |
| [16] | 2014 | HP   | \( n = 2 \) |
| [17] | 2015 | HP   | \( n \in \{2, 3, 5, 10\} \) |
| [33] | 2015 | HP   | \( n \in \{2, 5, 10, 20\} \) |
| [18] | 2016 | HP   | \( n \in \{2, 3, 5, 10\} \) |
| [2]  | 2016 | PIAC | \( n \in \{2, 3, 5, 10\} \) |
| [4]  | 2017 | PIAC | train: \( n \in \{2, 4, 5, 8, 10, 16, 20, 32, 40, 64\} \) |
| [7]  | 2019 | HP   | \( n \in \{2, 3, 5\} \) |
| [19] | 2019 | AS   | \( n \in \{2, 3, 5, 10\} \) |
| [13] | 2020 | PP   | \( n = 5 \) |
| [39] | 2020 | HP   | \( n = 5 \) |
| [8]  | 2020 | HP   | \( n = 5 \) |

Exploratory landscape analysis (ELA) [25, 26] provides a set of numerical low-level features based on a small sample of solutions. Unlike traditional analysis methods (e.g., FDC [15] and evolvability [43]), most ELA feature values are not human-interpretable [20]. In the ELA approach, the extracted features are used to characterize a fitness landscape of a black-box optimization problem by machine learning. As shown in Table 1, the ELA approach has been successfully applied to various tasks.

Table 1 shows the dimension \( n \) of the BBOB functions [11] in each previous study. This paper denotes the noiseless BBOB functions as the BBOB functions. As seen from Table 1, most previous studies focused only on low- and moderate-dimensional problems, typically with \( n \leq 20 \). Thus, the scalability of the ELA approach for large-scale optimization has been poorly understood. This limits the applicability of the ELA approach. Although large-scale real-world problems can be found in a wide range of research areas (e.g., [9, 31]), a rule of thumb is not available. Note that this situation is not unique to ELA. Except for [29], most previous studies of fitness landscape analysis for black-box numerical optimization (e.g., [28, 30, 41]) focused only on relatively low-dimensional problems.

Contributions. In this context, first, we investigate the computation time of features in the R-package flacco [20], which currently provides 17 feature classes. We use the BBOB function set and its large-scale version [48], which consists of the 24 functions with \( n \in \{20, 40, 80, 160, 320, 640\} \). The computation time of the ELA

\(^1\)A clear exception is [4]. For the sake of validation, \( n \) was different in the training and testing phases. The BBOB functions were used only for the training phase. The test functions with \( n = 100 \) were used to validate a performance prediction model, rather than to investigate the scalability of the ELA approach. The results in [4] showed that the performance prediction model does not work well when \( n \) in the testing phase is much larger than \( n \) in the training phase (i.e., \( n = 100 \)).
features has been paid little attention in the literature. While some previous studies (e.g., [5, 7, 18]) used the terms “computationally cheap/expensive” to represent a necessary sample size for ELA, this paper uses the terms only to represent the wall-clock time for computing features. In [20], the computation time of the 17 feature classes in fLacco was investigated on a problem with \( n = 2 \). In contrast, we analyze the influence of \( n \) on the computation time of some feature classes. The el1a_level and el1a_meta feature classes are one of the original six ELA feature classes [25]. Some previous studies (e.g., [13, 17–19, 25]) also reported their importance for various tasks. However, we demonstrate that the el1a_level and el1a_meta features are not available for \( n \geq 320 \) due to their high computational cost. Apart from the computational cost, as pointed out in [20], the four cell mapping feature classes (cm_angle, cm_conv, cm_grad, and gcm) [16] can be applied only to small-scale problems due to their properties. Thus, not all the 17 feature classes in fLacco are available for large-scale optimization.

To improve the scalability of the ELA approach, we propose a dimensionality reduction framework that computes features in a reduced \( m \)-dimensional space instead of the original \( n \)-dimensional solution space, where \( m < n \) (e.g., \( m = 2 \) and \( n = 640 \)). The proposed framework is inspired by dimensionality reduction strategies in Bayesian optimization [12, 38, 51]. In this paper, we use the weighting strategy-based principal component analysis (PCA) procedure in PCA-assisted Bayesian optimization (PCA-BO) [38] for dimensionality reduction. Since the proposed framework computes features in a reduced lower-dimensional space \( \mathbb{R}^m \) than the original solution space \( X \subseteq \mathbb{R}^n \), it can drastically reduce the computation time. In addition to the computational cost reduction, the proposed framework can make the cell mapping feature classes scalable for large-scale optimization. We evaluate the effectiveness of the proposed framework for predicting the high-level properties of the 24 BBOB functions with up to 640 dimensions.

**Related work.** The proposed framework is the first attempt to compute features in a reduced \( m \)-dimensional space in the field of black-box numerical optimization. We emphasize that dimensionality reduction is performed in the solution space, not the feature space. While some previous studies (e.g., [8, 34, 41, 50]) applied PCA [42] to the feature space for the sake of visualization, we apply PCA to the solution space. In the field of combinatorial optimization, some previous studies (e.g., [44, 49]) applied dimensionality reduction methods to the solution space for the sake of visualization. In contrast, we are not interested in such a visualization.

Belkhir et al. proposed a surrogate-assisted framework that computes features based on a small-sized sample \( X \) [3]. In their framework, an augmented solution \( X_{\text{aug}} \notin X \) is evaluated by a surrogate model \( M : \mathbb{R}^n \rightarrow \mathbb{R} \) instead of the actual objective function \( f \). Then, \( X_{\text{aug}} \) is added to an augmented sample \( X_{\text{aug}} \). Finally, features are computed based on the union of \( X \) and \( X_{\text{aug}} \). Their framework and our framework are similar in that the features are not computed based only on the original sample \( X \). However, their framework does not aim to reduce the dimension of the solution space.

We are interested in a speed-up technique at the algorithm level, not the implementation level. One may think that the scalability issue can be addressed by reimplementing fLacco in any compiled language (e.g., C). However, it is not realistic to reimplement fLacco without useful R libraries (e.g., mlr). Reimplementing fLacco is also not a fundamental solution for the scalability issue.

**Outline.** Section 2 provides some preliminaries. Section 3 investigates the computation time of ELA features. Section 4 explains the proposed framework. Section 5 explains our experimental setting. Section 6 shows our analysis results. Section 7 concludes this paper.

**Code availability.** The source code of the proposed framework is available at https://github.com/ryojitanabe/ela_drframework.

**Supplementary file.** Throughout this paper, we refer to a figure and a table in the supplementary file (https://ryojitanabe.github.io/pdf/gecco21-supp.pdf) as Figure S.* and Table S.*, respectively.

## 2 PRELIMINARIES

Section 2.1 explains the BBOB functions and their high-level properties. Section 2.2 describes ELA. Section 2.3 explains PCA-BO [38].

### 2.1 BBOB function set

The BBOB function set [11] consists of the 24 functions to evaluate the performance of optimizers in terms of difficulties in real-world black-box optimization. The functions \( f_1, ..., f_24 \) are grouped into the following five categories: separable functions \( (f_1, ..., f_5) \), functions with low or moderate conditioning \( (f_6, ..., f_9) \), functions with high conditioning and unimodal \( (f_{10}, ..., f_{14}) \), multimodal functions with good global structure \( (f_{15}, ..., f_{19}) \), and multimodal functions with weak global structure \( (f_{20}, ..., f_{24}) \). Each BBOB function further consists many instances. The large-scale version of the BBOB function set [48] provides problems with up to \( n = 640 \).

Table 2 shows the following seven high-level properties of the 24 BBOB functions [25]: multimodality, global structure, separability, variable scaling, search space homogeneity, basin sizes, and global to local optima contrast. For each BBOB function, the degree of a high-level property was categorized by experts, e.g., “none”, “low”, “medium”, and “high” for multimodality. Since some high-level properties presented in [25] were inappropriately classified, we slightly revised them in this work. As in [17], we classified the global structure property of the 13 unimodal functions as “strong”. Since the Lunacek bi-Rastrigin function \( f_{19} \) clearly has a multi-funnel structure, we classified its global structure property as “deceptive”. As in [27], we revised the separability property of \( f_3, f_6, \) and \( f_7 \). According to the definitions in [11], we revised the variable scaling and global to local optima contrast properties of some functions.

### 2.2 ELA

ELA [25] produces a set of numerical features based on a set of \( l \) solutions \( X = \{x_i\}_{i=1}^l \), where \( X \) is called the initial sample. In general, \( X \) is randomly generated in the \( n \)-dimensional solution space \( X \subseteq \mathbb{R}^n \) by a sampling method (e.g., Latin hypercube sampling). Then, the objective value \( f(x) \) is calculated for each solution \( x \in X \). Let \( \mathcal{Y} = \{f(x_1)\}_{i=1}^l \). We denote a pair of \( X \) and \( \mathcal{Y} \) as a data set \( \mathcal{D} \). Finally, an ELA feature maps \( \mathcal{D} \) to a numerical value.

The R-package fLacco [20] is generally used to compute features. Currently, fLacco provides 17 feature classes. Each feature class consists of more than one feature. For example, the ela_distr class provides the five features: skewness, kurtosis, number_of_peaks, costs_fun_evals, and costs_runtime.
Table 2: High-level properties of the 24 BBOB functions. The properties revised from [26] are highlighted.

| Function | Multin. | GL-struc. | Separ. | Scaling | Homog. | Basin | GL-cont. |
|----------|---------|-----------|--------|---------|--------|-------|---------|
| f1       | strong  | high      | none   | high    | none   | none  | none    |
| f2       | strong  | high      | none   | high    | none   | none  | none    |
| f3       | high    | strong    | high   | low     | high   | low   | low     |
| f4       | high    | strong    | high   | low     | high   | med.  | low     |
| f5       | high    | strong    | high   | low     | high   | med.  | low     |
| f6       | none    | strong    | high   | low     | high   | none  | none    |
| f7       | none    | strong    | none   | low     | none   | low   | low     |
| f8       | low     | strong    | none   | low     | med.   | low   | low     |
| f9       | low     | strong    | none   | low     | med.   | low   | low     |
| f10      | none    | strong    | none   | high    | none   | none  | none    |
| f11      | none    | strong    | none   | high    | none   | none  | none    |
| f12      | none    | strong    | none   | high    | none   | none  | none    |
| f13      | none    | strong    | none   | med.    | med.   | none  | none    |
| f14      | none    | strong    | none   | med.    | med.   | none  | none    |
| f15      | high    | strong    | none   | low     | high   | low   | low     |
| f16      | high    | med.      | none   | med.    | med.   | low   | low     |
| f17      | high    | med.      | none   | med.    | med.   | high  | low     |
| f18      | high    | strong    | none   | med.    | med.   | high  | low     |
| f19      | high    | strong    | none   | med.    | med.   | high  | low     |
| f20      | med.    | dec.      | none   | low     | high   | low   | high    |
| f21      | med.    | none      | none   | med.    | med.   | low   | low     |
| f22      | low     | none      | none   | high    | high   | med.  | high    |
| f23      | low     | none      | none   | high    | med.   | high  | high    |
| f24      | high    | dec.      | none   | low     | high   | low   | low     |

Table 3: 14 feature classes provided by flacco, except for elα_conv, elα_curv, and elα_local.

| Feature class | Name                      | Num. features |
|---------------|---------------------------|---------------|
| ela_distr     | y-distribution            | 5             |
| ela_level     | levelset                  | 20            |
| ela_meta      | meta-model                | 11            |
| nbc [17]      | nearest better clustering (NBC) | 7             |
| disp [17]     | dispersion                | 18            |
| ic [33]       | information content       | 7             |
| basic [20]    | basic                     | 15            |
| limo [20]     | linear model              | 14            |
| pca [20]      | principal component analysis | 10         |
| cm_angle [16] | cell mapping angle        | 10            |
| cm_conv [16]  | cell mapping convexity    | 6             |
| cm_grad [16]  | cell mapping gradient homog. | 6             |
| gcm [16]      | generalized cell mapping  | 75            |
| bt [20]       | barrier tree              | 90            |

Table 3 shows 14 feature classes provided by flacco. The original six ELA feature classes [25] are elα_conv, elα_curv, elα_local, ela_distr, ela_level, and ela_meta. As in most recent studies (see Table 1), we do not consider elα_conv, elα_curv, and elα_local. This is because they require additional function evaluations independently from the initial sample X.

The elα_level features are with regard to the distribution of Y. The elα_level class splits D into binary classes based on a pre-defined threshold value. Then, three classifiers (LDA, QDA, and MDA) are used to classify an objective value f(x) ∈ Y. Each elα_level feature represents the mean classification errors of a classifier over a 10-fold cross-validation. The ela_meta class fits linear and quadratic regression models to ela_meta, ela_level, and ela_curv. As in most recent studies (see Table 1), we do not consider elα_conv, elα_curv, and elα_local. This is because they require additional function evaluations independently from the initial sample X.

2.3 PCA-BO

Bayesian optimization is an efficient sequential model-based approach for computationally expensive optimization [40]. The Gaussian process regression (GPR) model is generally used in Bayesian optimization. However, it is difficult to apply the GPR model to large-scale problems. This is because the GPR model requires high computational cost as the dimension increases. One promising way to address this issue is the use of dimensionality reduction techniques. A similar approach has been adopted in the field of evolutionary computation (e.g., [21]).

PCA-assisted Bayesian optimization (PCA-BO) [38] uses PCA to reduce the original dimension n to a lower dimension m (m < n). PCA-BO also uses a weighting strategy to incorporate the information about the objective values into solutions so that a better solution is treated as more important than other solutions.

At the beginning of each iteration, PCA-BO re-scales a set of l solutions found so far X = {x_i}^l_{i=1} by the weighting strategy. First, all the solutions in X are ranked based on their objective values. For i ∈ {1, ..., l}, a weight value w_i is assigned to the i-th solution x_i ∈ X based on its rank r_i as follows: w_i = w_l / \sum_{j=1}^{l} w_j, where w_l = ln(l) / ln(l_i). Then, x_i is re-scaled as follows: x_i = x_l / (x_i - m), where m = 1 / l \sum_{x \in X} x is the mean vector of X. After the re-scaled version X of X is obtained by the weighting strategy, PCA-BO applies PCA to X. The dimensionality of each point in X is reduced from n to m. In other words, PCA maps an n-dimensional point x ∈ R^n to an m-dimensional point \hat{x} ∈ R^m. Then, PCA-BO fits a GPR model to \hat{X} = {\hat{x}_i}^l_{i=1} and searches a candidate solution that maximizes the acquisition function in R^m.
3 COMPUTATIONAL COST ISSUE IN ELA

Here, we investigate the computation time of the nine non-cell mapping ELA feature classes in Table 3 on problems with up to \( n = 640 \). Since the cell mapping features can be computed only for small dimensions, we do not consider them in this section.

Apart from the features, we point out that the improved Latin hypercube sampling method (IHS) [1] in flacco is time-consuming for large dimensions. Our results show that IHS requires approximately 6.6 hours for sampling 50\(\times\)n solutions on a problem instance for \( n = 160 \). The procedure of IHS did not finish within one day for \( n \geq 320 \). For details, see Figure S.1. This is because IHS calculates the Euclidean distance between points every time IHS adds a new point. For this reason, we use Latin hypercube sampling (LHS) instead of IHS throughout this paper.

3.1 Experimental setup

We conducted all experiments on a workstation with an Intel(R) 40-Core Xeon Gold 6230 (20-Core\(\times\)2) 2.1GHz and 384GB RAM using Ubuntu 18.04. We used Python version 3.8 and R version 3.6.3. We used the Python interface of flacco (pflacco), which is available at https://github.com/Reiyan/pflacco. We believe that the time to call a pflacco function from pflacco is negligible when \( n \) is large enough. We used the BBOB function set [11] for \( n \in \{2, 3, 5, 10\} \) and its large-scale version [48] for \( n \in \{20, 40, 80, 160, 320, 640\} \). Both function sets are available in the COCO platform [10]. We performed 31 independent runs on the first instance of \( f_i \) to measure the average time for computing features. We used 1hs in the pyDOE package to generate the initial sample \( X \). As recommended in [18], we set \( |X| \) to 50\(\times\)n (i.e., \( l = 50 \times n \) in Section 2.3). We did not take into account the time to generate \( X \) and evaluate their objective values \( Y \). Thus, we measured only the time to compute features.

3.2 Feature computation time in ELA

Figure 1 shows the average computation time of each feature class over 31 runs. We explain results of ela_level and ela_meta in Section 6.1. It seems that the computation time of all the feature classes (except for ela_distr) increases exponentially with respect to \( n \). Note that the computation time is influenced by both \( n \) and \( |X| \). Figure S.2 shows results when fixing \( |X| \) to 100. Figure S.3 shows results when fixing \( n \) to 2. Although setting \( |X| \) to a small constant number (i.e., \( |X| = 100 \)) can speed up the computation of features, the resulting features are likely to be ineffective.

As seen from Figure 1, the computation time of basic is lowest for \( n = 80 \). For \( n = 160 \), ela_distr is the fastest feature class in terms of the computation time, where the number of the ela_distr features is only five (see Table 3). Since the ela_distr features are based only on the objective values \( Y \), its computational cost depends only on \( |Y| \). Since limo and pca are based on relatively simple linear models, their computational cost is acceptable even for \( n = 640 \). The single-run computation of the limo and pca features took approximately 3.5 minutes and 1.1 minutes for \( n = 640 \), respectively. For \( n \geq 40 \), ic, nbc, and disp perform relatively similar in terms of the computation time. This is because the three feature classes calculate the distance between solutions in \( X \), where the distance calculation time depends on \( n \) and \( |X| \).

The computational cost of the ela_meta features is relatively low for \( n \leq 20 \), but it increases drastically for \( n \geq 40 \). This is mainly because fitting a quadratic model in ela_meta is time-consuming for large dimensions. Clearly, the computation of ela_level is the most expensive for any \( n \). This is because the ela_level features are based on results of the three classifiers in a 10-fold cross-validation as explained in Section 2.2. As shown in Figure 1, the single-run computation of the ela_meta and ela_level features took approximately 1.1 hours and 1.5 hours for \( n = 160 \), respectively. We could not measure the computation time of the ela_level and ela_meta features for \( n \geq 320 \) because their single-run computation for \( n = 320 \) did not finish within 3 days.

4 PROPOSED FRAMEWORK

Our results in Section 3 showed that the two important feature classes (ela_level and ela_meta) are not available for large-scale optimization due to their time-consuming process. Apart from that, as pointed out in [20], the cell mapping feature classes can be applied only to low-dimensional problems. To improve the scalability of these feature classes, we propose a simple dimensionality reduction framework. Let \( X = \{x_i\}_{i=1}^l \) be a set of \( l \) \( n \)-dimensional solutions, i.e., the initial sample. Let also \( Y = \{f(x_i)\}_{i=1}^l \) be a set of \( l \) objective values. As explained in Section 2.2, the features are computed based on \( X \) and \( Y \). However, as discussed above, some features cannot be computed in a practical time when \( n \) is large.

One simple way of addressing this issue is to reduce the original dimension \( n \) to a lower dimension \( m \) in dimensionality reduction strategies in Bayesian optimization. Previous studies (see Section 2.3) show that a dimensionality reduction strategy can effectively reduce the computational cost for fitting a surrogate model for large-scale optimization. Based on these promising results, we suppose that some (not all) features computed in the reduced \( m \)-dimensional space can substitute for their original versions.

In the proposed framework, first, a dimensionality reduction method is applied to \( X \). As a result, each solution \( x \in \mathbb{R}^n \) in \( X \) is mapped to a point \( \hat{x} \in \mathbb{R}^m \), where \( m < n \). Let \( \hat{X} = \{\hat{x}_i\}_{i=1}^l \) be a set of \( l \) \( m \)-dimensional points transformed from \( X \). For \( i \in \{1, \ldots, l\} \), the objective value \( f(x_i) \) is assigned to \( \hat{x}_i \) without any change, i.e., \( \hat{Y} := Y \). Thus, the proposed framework does not require additional
function evaluations. Finally, features are computed based on $\hat{X}$ and $\mathcal{Y}$ instead of $X$ and $\mathcal{Y}$.

Although any dimensionality reduction method can be integrated into the proposed framework, this paper uses the weighting strategy-based PCA approach in PCA-BO [38] (see Section 2.3). PCA is a simple linear transformation technique, but it is one of the most representative dimensionality reduction methods [47]. For this reason, PCA is a reasonable first choice. Note that the PCA procedure in PCA-BO is independent from the pca feature class.

One advantage of the proposed framework is that it can speed up the computation of time-consuming features since the features are computed in a reduced $m$-dimensional space. Another advantage is that the proposed framework can make the cell mapping feature classes scalable for large-scale optimization by setting $m$ to a small value (e.g., $m = 2$). One disadvantage is that features computed by the proposed framework may be misleading. This is because it is impossible for the proposed framework to extract all information about the original $n$-dimensional problem. For this reason, it is expected that the original features are more effective than their dimensionality reduction versions when they are available.

### 5 EXPERIMENTAL SETUP

This section explains the experimental setup. Unless explicitly noted, the computational environment is the same as in Section 3.1. As in most previous studies (e.g., [7, 16, 18, 25]), we used an off-the-shelf random forest classifier [6] for predicting the seven high-level properties of the 24 BBOB functions (see Table 2). We aim to examine the effectiveness of a feature set for the high-level property classification task rather than to achieve high accuracy by using a state-of-the-art classifier (e.g., deep neural network). We employed the scikit-learn implementation of random forest [35]. We set the number of trees to 1,000. According to the recommendation in [18], we set the size of the initial sample $|X| (= l)$ to $50 \times n$.

Below, Section 5.1 explains feature sets investigated in Section 6. Section 5.2 describes the cross-validation procedure in this work.

#### 5.1 Feature sets

Table 4 shows five feature sets investigated in this work. For a feature class computed by the proposed dimensionality reduction framework, we add a prefix “d_” to its original name. For dimensionality reduction, we employed the implementation of PCA-BO provided by the authors of [38] (https://github.com/wangronin/Bayesian-Optimization). We set the reduced dimension $m$ to 2.

Our results in Section 3.2 show that the following seven feature classes are relatively computationally cheap for large dimensions: ela_distr, basic, ic, disp, nbc, pca, and limo. We denote a set of the seven computationally cheap feature classes as C7. We consider C7 as a base-line feature set for analysis. Morgan and Gallagher demonstrated that the performance of the dispersion metric [22] for large dimensions ($n \leq 200$) can be improved by normalization [29]. One may think that the disp features with normalization is more effective for large dimensions. However, our preliminary results showed that normalization does not strongly affect the disp features. This is mainly because the disp features are already normalized by the mean or the median of the dispersion value of the whole sample $X$. Of course, further investigation is needed.

#### Table 4: Five feature sets investigated in this work.

| Name  | Feature classes                                                                 |
|-------|---------------------------------------------------------------------------------|
| C7    | {ela_distr, basic, ic, disp, nbc, pca, limo}                                     |
| C7-E2 | C7 U {ela_level, ela_meta}                                                      |
| C7-D2 | C7 U {d_ela_level, d_ela_meta}                                                  |
| C7-C4 | C7 U {gcm, cm_angle, cm_conv, cm_grad}                                           |
| C7-D4 | C7 U {d_gcm, d_cm_angle, d_cm_conv, d_cm_grad}                                  |

C7-E2 is a C7 with the two computationally expensive feature classes (ela_level and ela_meta), which can be computed only for $n \leq 160$ due to their time-consuming process. C7-D2 is a C7 with the dimensionality reduction versions of the two expensive feature classes. We analyze the effectiveness of the proposed framework by comparing C7-D2 with C7 and C7-E2.

C7-C4 is a C7 with the four cell mapping feature classes, which are available only for $n \leq 5$. Since bt can be computed only for $n = 2$, we did not use bt. C7-D4 is a C7 with the dimensionality reduction versions of the four cell mapping feature classes. Before computing the cell mapping features, we normalized each point $\hat{x}$ in $\hat{X}$ into the range $[0, 1]^m$ using the minimum and maximum values in $\hat{X}$ for each dimension.

For each $n$, we deleted a feature that takes the same value on all instances (e.g., disp.costs.fun_evals and basic.dim). As in [4, 7, 39], we did not perform feature selection because it deteriorated the performance of a classifier in our preliminary experiment. A similar observation was reported in [14]. This may be due to a leave-one-problem-out cross-validation (LOPO-CV) [7, 16] (see the next section), where the prediction model is validated on an unseen function. Although an analysis of cross-validation methods is beyond the scope of this paper, an in-depth investigation is needed.

#### 5.2 Cross-validation procedure

The ultimate goal of high-level property classification is to predict a high-level property of an unseen real-world problem. Thus, we need to evaluate the unbiased performance of a classifier on unseen test problem instances. For this reason, we adopted LOPO-CV [7, 16]. In LOPO-CV, a 24-fold cross-validation is performed on the 24 BBOB functions with each dimension $n$. Let $I_i$ be a set of problem instances of a function $f_i$, where $i \in \{1, \ldots, 24\}$. Since each BBOB function consists of 15 instances in the COCO platform, $|I_i| = 15$ for any $f_i$. While we used only the first instance of $f_i$ to measure the wall-clock time in Section 3, we consider all the 15 instances of all the 24 BBOB functions in Sections 6.2 and 6.3. In the $i$-th fold, $I_i$ (15 instances) is used in the testing phase. Thus, $I_1 \cup \cdots \cup I_{24} \setminus I_i$ (23×15 = 345 instances) are used in the training phase. Since the 24 BBOB functions have totally different properties from each other (see Table 2), problem instances used in the training and testing phases are also different in LOPO-CV. For this reason, high-level property classification in LOPO-CV is very challenging.

We also evaluated the performance of a classifier by using a leave-one-instance-out cross-validation (LOIO-CV) [7]. In LOIO-CV, a 15-fold cross-validation is performed on the 15 problem instances. In the $i$-th fold, the 24 $i$-th problem instances are used in the testing phase. Thus, the remaining $14 \times 24 = 336$ instances are used in the training phase. Since problem instances used in the training
and testing phases are very similar, LOIO-CV is much easier than LOPO-CV. In fact, our preliminary results showed that some classifiers could perfectly predict some high-level properties of the 24 BBOB functions in LOIO-CV (i.e., accuracy = 1). For this reason, we consider only the more challenging LOPO-CV in this paper.

6 RESULTS

First, Section 6.1 demonstrates that the proposed framework can reduce the computational cost of the two expensive feature classes (ela_meta and ela_level) for large dimensions. Then, Section 6.2 examines the effectiveness of the five feature sets in Table 4 for predicting the high-level properties of the 24 large-scale BBOB functions. Finally, Section 6.3 analyzes the similarity between the original features and their dimensionality reduction versions.

For the sake of simplicity, we refer to "a random forest classification model using a feature set \( F \)" as \( \tilde{F} \). For example, we denote "the accuracy of a random forest classifier using C7-D2" as "the accuracy of C7-D2".

6.1 Computation time reduction

Figure 1 (see Section 3.2) shows the average time for computing the d_ela_level and d_ela_meta features. In Figure 1, the computation time includes the PCA-BO procedure for dimensionality reduction. Since the proposed framework cannot be applied to problems with \( n \leq m \), we do not show results of the d_ela_level and d_ela_meta features for \( n = 2 \). Figure S.4 shows results of the dimensionality reduction versions of the four cell mapping feature classes in C7-D4. They perform relatively similar to pca and d_ela_meta for \( n = 640 \).

As shown in Figure 1, the computation time of d_ela_level is slightly higher than that of ela_level for \( n \in \{3, 5\} \). This is because the dimensionality reduction procedure is more time-consuming than the feature computation procedure for small dimensions. In contrast, d_ela_level is much faster than ela_level in terms of the computation time as the dimension increases. For example, while the computation of ela_level requires approximately 1.5 hours for \( n = 160 \), that of d_ela_level requires only approximately 2.1 minutes. Notably, the computation of d_ela_level is cheaper than that of the three distance-based feature classes (ic, disp, and nbc) for \( n = 640 \). Results of d_ela_meta is similar to the results of d_ela_level discussed above. The computation of d_ela_meta is much cheaper than that of ela_meta for \( n \geq 40 \). Interestingly, d_ela_meta is faster than pca for \( n \geq 320 \) in terms of the computation time. As seen from Figure 1, the computation of d_ela_meta is faster than that of limo for \( n \geq 320 \). This result indicates that fitting a quadratic model in a reduced m-dimensional space can be faster than fitting a linear model in the original n-dimensional space as \( n \) increases. In summary, our results show that the proposed framework can effectively reduce the computation time of ela_meta and ela_level for large dimensions.

6.2 Effectiveness of features computed by the proposed framework

6.2.1 Results of C7, C7-E2, and C7-D2. Table 5(a)–(g) show the average accuracy of C7, C7-E2, and C7-D2 for the seven high-level properties, respectively. Table 5(h) also shows the overall average. In Table 5, "Na" means that the corresponding feature set is not available for a given \( n \) due to its time-consuming process. The best and second best data are highlighted by dark gray and gray, respectively. When only two data are available, only the best data is highlighted by dark gray. Note that we built one random forest model for each fold, each feature set, each classification task, and each dimension. Tables 5.1(a)–(g) also show the standard deviation of the accuracy. Due to LOPO-CV, the standard deviation is large.

As shown in Table 5, C7-E2 performs the best in terms of the accuracy in most cases for \( n \leq 160 \), except for the prediction of the homogeneity property. These results indicate the importance of the ela_meta and ela_level features to predict a high-level property of an unseen problem. However, C7-E2 is not available for \( n \geq 320 \) due to the high-computational cost of ela_meta and ela_level. In contrast, C7-D2 achieves better accuracy than C7 in most cases (especially for \( n \geq 320 \), except for the prediction of the separability property. These results indicate that the d_ela_meta and d_ela_level features can be substituted for their original versions for large-scale optimization. The d_ela_meta and d_ela_level features are particularly beneficial for predicting the multimodality and basin size properties (see Tables 5(a) and (f)). Since the proposed framework reduces the dimensionality of each point in the initial sample \( X \) from \( n \) to \( m \), it is unsurprising that the d_ela_meta and d_ela_level features do not provide meaningful information about separability. Consequently, for the prediction of the separability property, C7-D2 performs worse than C7 in terms of accuracy due to the misleading features (see Table 5(c)). As already discussed in Section 4, this is one disadvantage of the proposed framework. The influence of the dimension \( n \) on the performance of the ELA approach has not been well analyzed in the literature. Intuitively, it is expected that a prediction model performs poorly as the dimension increases. Roughly speaking, the results in Tables 5(a), (b), (c), and (d) are consistent with our intuition. In contrast, the results in Tables 5(e), (f), and (g) show that the accuracy of C7, C7-E2, and C7-D2 is improved as \( n \) increases. For example, the accuracy of C7 for predicting the homogeneity property is 0.633 and 0.772 for \( n = 2 \) and \( n = 640 \), respectively. This may be because the size of the initial sample \( X \) increases linearly with respect to \( n \), where we set \( |X| \) to 50 × \( n \). The large-sized \( X \) can possibly be beneficial to capture the degree of the three properties (homogeneity, basin size, and global to local optima contrast) independently from \( n \).

6.2.2 Results of C7, C7-C4, and C7-D4. Table 6 shows the overall average accuracy of C7, C7-C4, and C7-D4. Table S.2 shows detailed results. As seen from Table 6, C7-C4 performs the best in terms of accuracy for \( n \leq 5 \). Although the cell mapping features are available only for \( n \leq 5 \), the proposed framework can improve their scalability for \( n > 5 \). As shown in the results of C7-D4, the cell mapping features computed by the proposed framework are helpful for high-level property classification for \( n \geq 10 \). However, C7-D4 performs poorly for small dimensions similar to C7-D2. Thus, it is better to use the proposed framework only for large dimensions.

6.2.3 Feature importance. We investigate which features are particularly important in C7-D2 and C7-D4. Table 7 shows the average rankings of the features computed by the proposed framework for \( n = 640 \). Results for all dimensions are relatively similar. First, we
Table 5: Average accuracy of C7, C7-E2, and C7-D2 on the 24 BBOB functions with $n \in \{2, 3, 5, 10, 20, 40, 80, 160, 320, 640\}$.

| (a) Multimodality | (b) Global structure | (c) Separability | (d) Variable scaling |
|-------------------|----------------------|-----------------|----------------------|
| C7 | C7 | C7 | C7 | C7 | C7 | C7 | C7 |
| 2 | 0.642 | 0.703 | 0.581 | 0.617 | 0.677 | 0.578 | 0.651 | 0.622 | 2 | 0.756 | 0.805 | 0.700 | 0.725 | 2 | 0.581 | 0.586 |
| 3 | 0.569 | 0.597 | 0.578 | 0.628 | 0.781 | 0.669 | 0.583 | 0.689 | 3 | 0.789 | 0.856 | 0.786 | 0.692 | 3 | 0.611 | 0.583 |
| 5 | 0.536 | 0.625 | 0.622 | 0.769 | 0.772 | 0.761 | 0.597 | 0.622 | 5 | 0.753 | 0.864 | 0.763 | 0.614 | 5 | 0.642 | 0.600 |
| 10 | 0.522 | 0.631 | 0.647 | 0.708 | 0.750 | 0.692 | 0.608 | 0.697 | 10 | 0.758 | 0.828 | 0.753 | 0.599 | 10 | 0.508 | 0.556 |
| 20 | 0.531 | 0.729 | 0.639 | 0.714 | 0.542 | 0.697 | 0.539 | 0.578 | 20 | 0.708 | 0.806 | 0.744 | 0.539 | 20 | 0.539 | 0.578 |
| 40 | 0.556 | 0.689 | 0.617 | 0.711 | 0.711 | 0.741 | 0.676 | 0.836 | 40 | 0.767 | 0.836 | 0.758 | 0.676 | 40 | 0.539 | 0.556 |
| 80 | 0.600 | 0.783 | 0.622 | 0.697 | 0.697 | 0.697 | 0.703 | 0.775 | 80 | 0.703 | 0.775 | 0.686 | 0.802 | 80 | 0.542 | 0.583 |
| 160 | 0.522 | 0.650 | 0.614 | 0.647 | 0.500 | 0.631 | 0.676 | 0.725 | 0.667 | 160 | 0.697 | 0.725 | 0.667 | 0.658 | 160 | 0.558 | 0.563 |
| 320 | 0.544 | 0.578 | 0.664 | 0.667 | 0.667 | 0.667 | 0.722 | 0.669 | 320 | 0.547 | 0.583 | 0.583 | 0.664 | 320 | 0.542 | 0.542 |
| 640 | 0.514 | 0.583 | 0.667 | 0.678 | 0.678 | 0.678 | 0.725 | 0.669 | 640 | 0.542 | 0.542 | 0.542 | 0.664 | 640 | 0.542 | 0.542 |

Table 6: Overall average accuracy of C7, C7-C4, and C7-D4 for predicting the seven high-level properties of the BBOB functions.

| (a) Homogeneity | (b) Basin size | (c) Global to local optima cont. | (d) Overall average |
|------------------|----------------|-------------------------------|-------------------|
| C7 | C7-C4 | C7-D2 | C7 | C7 | C7-D2 | C7 | C7 | C7-D2 |
| 2 | 0.633 | 0.638 | Na | 2 | 0.450 | 0.492 | Na | 2 | 0.572 | 0.642 | Na |
| 3 | 0.613 | 0.621 | 0.608 | 3 | 0.554 | 0.547 | 0.522 | 3 | 0.586 | 0.608 | 0.572 |
| 5 | 0.647 | 0.578 | 0.474 | 5 | 0.508 | 0.628 | 0.522 | 5 | 0.497 | 0.669 | 0.536 |
| 10 | 0.722 | 0.658 | 0.769 | 10 | 0.494 | 0.608 | 0.636 | 10 | 0.542 | 0.656 | 0.619 |
| 20 | 0.592 | 0.731 | 0.731 | 20 | 0.467 | 0.647 | 0.553 | 20 | 0.544 | 0.728 | 0.617 |
| 40 | 0.761 | 0.700 | 0.744 | 40 | 0.531 | 0.617 | 0.614 | 40 | 0.533 | 0.708 | 0.597 |
| 80 | 0.703 | 0.639 | 0.742 | 80 | 0.464 | 0.650 | 0.597 | 80 | 0.558 | 0.703 | 0.619 |
| 160 | 0.656 | 0.625 | 0.683 | 160 | 0.472 | 0.578 | 0.614 | 160 | 0.555 | 0.636 | 0.622 |
| 320 | 0.719 | 0.761 | Na | 320 | 0.408 | 0.608 | Na | 320 | 0.550 | 0.647 | Na |
| 640 | 0.772 | 0.781 | Na | 640 | 0.456 | 0.506 | Na | 640 | 0.572 | 0.674 | Na |

ranked the features based on their impurity-based feature importance values computed by the random forest for each fold, where we used the feature importances_attribute of scikit-learn. Then, we calculated the average rankings of the features over the 24-folds and all seven classification tasks for each dimension. Table 7 shows only the results of top 10 features. Tables S3–S6 show the results of all features for all dimensions.

As seen from Table 7(a), the seven d_eia_meta features are ranked as more important than the d_eia_level features. For this reason, one may think that the d_eia_level features can be removed from C7-D2. However, we observed that removing the d_eia_level features from C7-D2 degrades its performance. As shown in Table 7(b), the d_cm_angle features are more important than the other cell mapping features. In contrast to C7-D2, all the cell mapping features always rank low. For example, the average ranking of d_cm_angle.y_ratio_best2worst.sd is 43.2. However, as demonstrated in Section 6.2.2, the cell mapping features computed by the proposed framework are beneficial for predicting the high-level properties of the BBOB functions. This result indicates the difficulty of feature selection in the ELA approach.

6.2.4 Analysis of m in the proposed framework. The reduced dimension $m$ can be viewed as a control parameter in the proposed framework. Although we set $m$ to 2 in this section, we here investigate the influence of $m$ on the performance of the ELA approach.

Table 7 shows results of C7-D2 and C7-D4 with different $m$. Tables S7 and S8 show detailed results. Since the cell mapping features can be computed for $2 \leq n \leq 5$, we set $m$ to 2, 3, and 5 for C7-D4. Table 8(a) and (b) indicate that a suitable $m$ depends on $n$ and a feature set. Roughly speaking, for C7-D2, $m = 2$ is suitable for $n \leq 320$, but $m = 10$ is the best setting for $n = 640$. Based on this observation, we suggest the use of $m = 2$ for C7-D2, but it may be better to set $m$ to a large value (e.g., $m \in \{5, 10\}$) for $n \geq 640$. For C7-D4, it is difficult to give a general conclusion, but any $m \in \{2, 3, 5\}$ can be a good first choice.

6.3 Similarity of feature values

We analyze the similarity between features and their dimensionality reduction versions. Figure 2 shows values of the ela_meta.lin_simple.coef.max feature and its dimensionality reduction version
Table 8: Overall average accuracy of C7-D2 and C7-D4 with different m for n ∈ {5, 10, 20, 40, 80, 160, 320, 640}.

| (a) C7-D2 | (b) C7-D4 |
|-----------|-----------|
|           |           |
| 2         | 3         | 5         | 10        | 2         | 3         | 5         |
| 5         | 0.640     | 0.635     | 0.639     | Na        | 0.640     | 0.635     | 0.639     | Na        |
| 10        | 0.654     | 0.670     | 0.645     | 0.643     | 0.628     | 0.623     | 0.641     | Na        |
| 20        | 0.655     | 0.633     | 0.642     | 0.652     | 0.623     | 0.616     | 0.649     | 0.621     |
| 40        | 0.656     | 0.637     | 0.647     | 0.652     | 0.632     | 0.633     | 0.653     | 0.635     |
| 80        | 0.639     | 0.643     | 0.646     | 0.635     | 0.618     | 0.630     | 0.627     | 0.635     |
| 160       | 0.617     | 0.623     | 0.620     | 0.626     | 0.593     | 0.596     | 0.601     | 0.604     |
| 320       | 0.621     | 0.631     | 0.629     | 0.628     | 0.600     | 0.605     | 0.602     | 0.604     |
| 640       | 0.626     | 0.628     | 0.620     | 0.638     | 0.614     | 0.613     | 0.611     | 0.614     |

**(m = 2)** on 15 instances of \(f_1, f_6, f_{10}, f_{15}, \text{ and } f_20\) with \(n = 160\). We selected them from the five function groups, respectively. We also selected \(\text{ela}_\text{meta} \_\text{lin} \_\text{simple} \_\text{coef} \_\text{max}\) based on the results in Section 6.2.3. Figures S.5–S.130 show values of the other features in C7-D2 and C7-D4. As seen from the scale of the y-axis in Figures 2(a) and (b), values of the original feature and its dimensionality reduction version are totally different. The cause of this observation may be the PCA transformation. An analysis of the proposed framework with nonlinear transformation techniques (e.g., t-SNE [46]) is an avenue for future work. However, the relative rankings of the feature values on \(f_1, f_6, f_{10}, f_{15}, \text{ and } f_20\) are similar in Figures 2(a) and (b). For example, the two features take the smallest and largest values on \(f_1\) and \(f_20\), respectively.

Figure 3 shows the Kendall rank correlation coefficient \(\tau\) values of the 8 \(\text{ela}_\text{meta} \_\text{and} \_\text{d}_\text{ela}_\text{level}\) features on the 24 BBOB functions with \(n = 160\). Since 3 out of the 11 \(\text{ela}_\text{meta}\) and \(\_\text{d}_\text{ela}_\text{meta}\) features include non-unique values, we could not calculate their \(\tau\) values. Let \(a = (a_1, ..., a_{24})^\top\) and \(b = (b_1, ..., b_{24})^\top\) be the values of the original feature and its dimensionality reduction version on the 24 BBOB functions respectively, where \(a_i \in a \text{ and } b_i \in b\) are the average over 15 instances of \(f_i\) \((i \in \{1, ..., 24\})\). The \(\tau\) value quantifies the similarity of the rankings of \(a\) and \(b\). Here, \(\tau\) takes a value in the range \([-1, 1]\). If the rankings of \(a\) and \(b\) are perfectly consistent, \(\tau = 1\). If they are perfectly inconsistent, \(\tau = -1\). If they are perfectly independent, \(\tau = 0\).

As shown in Figure 3, \(\text{ela}_\text{meta} \_\text{lin} \_\text{simple} \_\text{coef} \_\text{max}\) is strongly consistent with \(\text{d}_\text{ela}_\text{meta} \_\text{lin} \_\text{simple} \_\text{coef} \_\text{max}\) \((\tau = 0.96)\). We can see high \(\tau\) values for the following three features: \(\text{lin} \_\text{simple} \_\text{adj} \_\text{r2}, \text{lin} \_\text{simple} \_\text{intercept}, \text{and} \_\text{lin} \_\text{simple} \_\text{coef} \_\text{min}\). These results indicate that the relative rankings of values of some features and their dimensionality reduction versions can possibly be similar even if their absolute values are different. We believe that a feature computed by the proposed framework can substitute for its original version when their relative rankings are similar.

Note that this is not always the case. Since \(\tau\) is close to zero for \(\text{lin} \_\text{simple} \_\text{coef} \_\text{max} \_\text{by} \_\text{min}, \text{quad} \_\text{simple} \_\text{cond}, \text{and} \_\text{costs} \_\text{runtime}\) (see (c), (g), and (h) in Figure 3), their original and dimensionality reduction versions are almost independent. Figures S.131–S.135 show results of the other features. The results of \(\text{d}_\text{ela}_\text{level}\) and \(\text{d}_\text{gcm}\) are relatively similar to Figure 3. In contrast, we observed that \(\tau\) is close to zero for most features in the \(\text{d}_\text{cm} \_\text{angle}, \text{d}_\text{cm} \_\text{grad}, \text{and} \_\text{d}_\text{cm} \_\text{conv}\) classes. It would be better to consider that the dimensionality reduction versions of these features are not related to their original versions.

7 CONCLUSION

First, we pointed out the computational cost issue in the ELA features for large dimensions. Our results revealed that the two important feature classes \(\text{ela}_\text{level}\) and \(\text{ela}_\text{meta}\) cannot be applied to large-scale optimization due to their high computational cost. To address this issue, we proposed the dimensionality reduction framework that computes features in a reduced low-dimensional space. Our results show that the proposed framework can drastically reduce the computation time of \(\text{ela}_\text{level}\) and \(\text{ela}_\text{meta}\). Our results also show the effectiveness of features (including the four cell mapping features) computed by the proposed framework on the BBOB functions with up to 640 dimensions. In addition, we found that the relative rankings of values of some (not all) features and their dimensionality reduction versions are similar.

We focused on extending the existing features for large dimensions, but it is promising to design a new computationally cheap feature. An extension of landscape-aware algorithm selection methods for large-scale optimization is an avenue for future work. It is not obvious if the classification accuracy shown in this paper is good enough for practical purposes. There is room for discussion about the benchmarking methodology of the ELA approach.

ACKNOWLEDGMENTS

This work was supported by Leading Initiative for Excellent Young Researchers, MEXT, Japan.
Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and L. Darrell Whitley (Eds.). Springer, 411–423. https://doi.org/10.1007/978-3-319-99529-4_39

[32] Mario A. Muñoz, Michael Kirley, and Suman K. Halgamuge. 2012. A Meta-learning Prediction Model of Algorithm Performance for Continuous Optimization Problems. In Parallel Problem Solving from Nature - PPSN XII - 12th International Conference, Taormina, Italy, September 1-5, 2012. Proceedings, Part I (Lecture Notes in Computer Science, Vol. 7491), Carlos A. Coello Coello, Vincenzo Cutello, Kayanmoy Deb, Stephanie Forrest, Giuseppe Nicosa, and Mario Pavone (Eds.). Springer, 226–255. https://doi.org/10.1007/978-3-642-32957-1_23

[33] Mario A. Muñoz, Michael Kirley, and Suman K. Halgamuge. 2015. Exploratory Landscape Analysis of Continuous Space Optimization Problems Using Information Content. IEEE Transactional Evolutionary Computation, 19, 1 (2015), 74–87. https://doi.org/10.1109/TEVC.2014.2302086

[34] Mario A. Muñoz and Kate Smith-Miles. 2015. Effects of function translation and dimensionality reduction on landscape analysis. In IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015. IEEE, 1336–1342. https://doi.org/10.1109/CEC.2015.7257043

[35] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Mathieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830. http://dl.acm.org/citation.cfm?id=2078195

[36] Erik Pitzer and Michael Affenzeller. 2012. A Comprehensive Survey on Fitness Landscape Analysis. In Recent Advances in Intelligent Engineering Systems, Janos C. Fodor, Ryszard Klemptous, and Carmen Paz Suarez Araujo (Eds.), Studies in Computational Intelligence, Vol. 378. Springer, 161–191. https://doi.org/10.1007/978-3-642-33229-9_8

[37] Mike Preuss. 2012. Improved Topological Niching for Real-Valued Global Optimization. In Applications of Evolutionary Computation - EvoApplications 2012: EvoCOMNET, EvoCOMPLEX, EvoFIN, EvoGAMES, EvoHOT, EvoASP, EvoNUM, EvoPAR, EvoRSK, EvoSTIM, and EvoSTOC, Malaga, Spain, April 11-15, 2012, Proceedings (Lecture Notes in Computer Science, Vol. 7248), Cecilia Di Chio, Alexandros Agapitos, Stefano Cagnoni, Carlos Cotta, Francisco Fernández de Vega, Gianni A. Di Caro, Rolf Drechsler, Andró Eikati, Anna Isabel Esparrac-Alicaz, Muddassar Farooq, William B. Langdon, Juan Julián Merelo Guervós, Mike Preuss, Hendrik Richter, Sara Silva, Anabela Simões, Giovanni Squillero, Ernesto Tarantino, Andrea Tetramanzu, Julian Togelius, Neil Urquhart, Sima Uyar, and Georgios N. Yannakakis (Eds.). Springer, 386–395. https://doi.org/10.1007/978-3-642-29178-4_39

[38] Elena Raponi, Hao Wang, Mariusz Bujny, Simonetta Boria, and Carola Doerr. 2020. High Dimensional Bayesian Optimization Assisted by Principal Component Analysis. In Parallel Problem Solving from Nature - PPSN XVII - 17th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12259), Thomas Bäck, Mike Preuss, André H. Deutz, Hao Wang, Carola Doerr, Michael T. M. Emmerich, and Heike Trautmann (Eds.), Springer, 169–183. https://doi.org/10.1007/978-3-030-58112-1_12

[39] Quentin Renau, Carola Doerr, Johann Dréo, and Benjamin Doerr. 2020. Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy. In Parallel Problem Solving from Nature - PPSN XII - 12th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12270), Thomas Bäck, Mike Preuss, André H. Deutz, Hao Wang, Carola Doerr, Michael T. M. Emmerich, and Heike Trautmann (Eds.), Springer, 159–153. https://doi.org/10.1007/978-3-030-58115-2_10

[40] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. Journal of Artificial Intelligence Research 55 (2016), 361–387. https://doi.org/10.1613/jair.4806

[41] Konstantinos Varelas, Ouassim Ait ElHara, Dimo Brockhoff, Nikolaus Hansen, Duc Manh Nguyen, Tea Tular, and Anne Auger. 2020. Benchmarking large-scale continuous optimizers: The bbob-large-scale testbed, a COCO software guide and beyond. Applied Soft Computing 97, Part (2020), 106737. https://doi.org/10.1016/j.asoc.2020.106737

[42] Nadarajan Vermaan and Gabriela Ochoa. 2018. Visualising the global structure of search landscapes: genetic improvement as a case study. Genetic Programming and Evolvable Machines, 19, 3 (2018), 317–349. https://doi.org/10.1007/s10710-018-9328-1

[43] Urban Škvorc, Tome Effimov, and Peter Korösec. 2020. Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. Applied Soft Computing 90 (2020), 106138. https://doi.org/10.1016/j.asoc.2020.106138

[44] Ziyu Wang, Frank Hutter, Masrouz Zoghi, David Matheson, and Nando de Freitas. 2016. Bayesian Optimization in a Billion Dimensions via Random Embeddings. Journal of Machine Learning Research 15 (2016), 4569–4594. https://doi.org/10.1109/JPROC.2015.2494218

[45] Shashank Marwah, Konstantinos Varelas, Ouassim Ait El Hara, Dimo Brockhoff, Nikolaus Hansen, Duc Manh Nguyen, Tea Tular, and Anne Auger. 2020. Benchmarking large-scale continuous optimizers: The bbob-large-scale testbed, a COCO software guide and beyond. Applied Soft Computing 97, Part (2020). 106737. https://doi.org/10.1016/j.asoc.2020.106737

[46] Lauren's van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. Journal of Machine Learning Research 9, 86 (2008), 2579–2605.

[47] Lauren's van der Maaten, Eric Postma, and Jaap van den Herik. 2008. Dimensionality Reduction: A Comparative Review. Technical Report. Tilburg University.