# New Features for Continuous Exploratory Landscape Analysis based on the SOO Tree

Bilel Derbel, Arnaud Liefooghe, Sébastien Verel, Hernan Aguirre, Kiyoshi Tanaka

# New Features for Continuous Exploratory Landscape Analysis based on the SOO Tree

Bilel Derbel
Univ. Lille, CNRS, Inria, CRIStAL
Lille, France
bilel.derbel@univ-lille.fr

Arnaud Liefooghe
Univ. Lille, CNRS, Inria, CRIStAL
Lille, France
arnaud.liefooghe@univ-lille.fr

Sébastien Verel
Univ. Littoral Côte d'Opale, UR 4491,
LISIC, Laboratoire d'Informatique
Signal et Image de la côte d'Opale
F-62100 Calais, France
verel@univ-littoral.fr

Hernán Aguirre
Shinshu University
Nagano, Japan
ahernan@shinshu-u.ac.jp

Kiyoshi Tanaka
Shinshu University
Nagano, Japan
tanaka@shinshu-u.ac.jp

## ABSTRACT

Extracting a priori knowledge informing about the landscape underlying an unknown optimization problem has been proved extremely useful for different purposes, such as designing finely-tuned algorithms and automated solving techniques. Focusing on continuous domains, substantial progress has been achieved with the development of the so-called exploratory landscape analysis (ELA) approach, which provides a unified methodology for integrating features into sophisticated machine learning techniques. In particular, much efforts have been devoted to the systematic design of algorithm selection models aiming at improving existing state-of-art solvers. Nonetheless, designing the ELA features themselves is a bottleneck that can prevent further advances. The contribution of this paper is thereby two fold. Firstly, we consider the design of insightful features on the basis of the search tree constructed by the so-called SOO global optimizer, which is shown to imply an informative sampling of the search space using a limited budget. Secondly, we provide empirical evidence on the relevance of the proposed features and their potential in complementing existing ELA features for both predicting high-level problem properties, and selecting algorithms from a portfolio of available solvers. Our empirical findings are based on a comprehensive analysis using the diverse set of BBOB functions and solvers from the COCO platform.

## KEYWORDS

Blackbox landscape features, exploratory landscape analysis.

## 1 INTRODUCTION

*General context.* We are concerned with pushing a step forward the development of a more unified use of blackbox optimization techniques and high-level search paradigms, coming eventually from various research fields in evolutionary computation, computational intelligence and machine learning. In fact, blackbox optimization refers to the situation where no problem-specific properties nor hypothesis is known beforehand. As such, nothing but the fitness/objective value associated with a given (candidate) solution can be used by the optimization process. Therefore, the efficient solving of a blackbox optimization problem can be challenging from different perspectives, and there exist no (universal) rule-of-thumb for it independently of the specific properties of the (unknown) problem being tackled. Given the large spectrum of available algorithms, components and parameters, it becomes more and more difficult, even for expert and well-trained algorithm designers, to identify the best possible choice when tackling a given instance of the same problem or when considering different problems coming from different application domains. In such a context, one interesting line of research is to develop general-purpose techniques that are able to extract and to inject blackbox landscape features into the optimization process, at the aim of gaining in generality and designing more powerful automated and autonomous algorithms [11, 18].

Since the work of Rice [33], where a feature extraction and a per-instance algorithm selection methodology was introduced, several studies having different complementary objectives have been conducted; e.g., [14, 22, 28, 37, 39]. The goal is not only to get rid of the burden of a manual calibration or the bias of ad-hoc decisions, but more importantly to set up a principled approach for algorithms' design, allowing to systematically explore their strengths and weaknesses when tackling a whole family of problem instances. Reviewing the abundant literature about the subject is out of the scope of this paper. However, it should be clear that such a challenging aim can only be achieved through a global inter-disciplinary approach integrating and mixing the progress made in parallel to address more fine-grained issues such as: defining informative landscape features [34], designing new high level algorithms and leveraging machine learning techniques to the blackbox optimization field [36], improving the robustness of (on-line and off-line) automatic configuration tools [20], contributing to the foundation of landscape-aware algorithm selection methods [4], etc.

*Objectives and related work.* In this paper, we continue the efforts of the community by considering the design of novel landscape features dedicated to single-objective continuous blackbox optimization. In fact, one can find a number of studies on the difficultly of solving a problem instance using a landscape-oriented approach, that is, by attempting to capture the characteristics of the search

*FOGA'19, August 27–29, 2019, Potsdam, Germany*
2019. ACM ISBN pre-print…$
https://doi.org/10.1145/3299904.3340308

space, such as its smoothness/ruggedness, peaks, etc., using numerical values extrapolated from the fitness values of some points sampled from the variable space. Continuous fitness landscape analysis is however a challenging task, especially because of the need of efficient sampling procedures. Meanwhile, much effort has been made to show the benefits of using informative problem-independent landscape features. The so-called Exploratory Landscape Analysis (ELA) approach considered in [24, 25], and developed in a number of subsequent studies [19], constitutes one major achievement made by the community in the last years. An ELA approach can be viewed as providing a unifying and sophisticated generic methodology combining feature extraction with supervised machine learning techniques, to perform advanced optimization tasks. These tasks range from predicting high-level global properties of a variety of optimization problems, to selecting an accurate and well-performing algorithm for unknown problem instances. In this respect, the recent development of the flacco R package [17] (feature-based landscape analysis of continuous and constrained optimization problems), providing a programming interface for extracting ELA features, can be considered as a significant achievement. In fact, the computation of the most relevant features is enabled in a user-friendly manner, and hence their smooth integration in the ELA developments. Nonetheless, one key ingredient for the success of an ELA approach relies on the ability to design and to extract meaningful landscape features. It should hence be clear that any advance in the design of novel informative features shall help the widespread uptake of such an approach. This is precisely the main goal of our work.

The landscape features that were integrated and experimented so far within the most recent state-of-the-art ELA techniques are based on the sampling of the variable space using Latin Hypercube Sampling (LHS), and its variants. Such initial LHS design are then used to compute a number of low-level statistics characterizing the problem at hand. These statistics are organized in different classes, ranging from the ones measuring the distribution of the objective function values, to those rendering the fitting quality of machine learning models, or some information about convexity and curvature, to mention a few. Within the flacco tool, more than 300 ELA numerical features are hence available.

*Methodology and contribution overview.* Most available ELA features are based on LHS, as the initial intent is to cover different regions of the variable space. Some feature classes may however require additional function evaluations. This is for example the case of the so-called Local Search-based class. Starting from an initial LHS design, a number of local searches can indeed be supplied. Such features inform about the relative number of local optima and their basin sizes. Although such features were initially intended to capture high-level properties of the landscape, they can intuitively inform about the difficulties that more advanced algorithms may face when tackling specific landscape. In other words, we view such features as conceptually interesting as they could correlate with the behavior of a concrete solving procedure. However, computing them can be computationally expensive, and their use does not seems to be advised as they are very often excluded from the latest ELA developments, i.e., only the cheap features that do not require further function evaluations are usually considered. The idea developed in this paper is specifically to derive novel cheap features that can complement the existing ELA features, while being specifically inspired by what a concrete algorithm can collect about the landscape. For this purpose, we propose to rely on a global search procedure called Simultaneous Optimistic Optimization (SOO) [27, 32]. The SOO algorithm is initially motivated by pure theoretical considerations, and is based on applying bandit theory to tackle global optimization. The algorithm has the remarkable properties of having theoretically provable performance and convergence guarantees under some mild assumptions. The key observation motivating our ELA approach is that, to obtain such strong guarantees, the SOO algorithm manages to sample the search space while finely tuning its exploration/exploration search ability. As such, it is able to provide a global view of the variable space within a very restricted budget, which makes it a serious candidate to perform landscape analysis. Pushing this idea deeper, we propose to leverage the SOO algorithm in order to extract a broad range of landscape features that can be integrated in an ELA approach.

More precisely, we propose new features grasping the characteristics of the search tree constructed by SOO. Based on a visual inspection, we show that the shape of the SOO tree search vary substantially across different functions. We hence propose a number of statistics informing about the shape of the tree and the fitness distribution of nodes within the different tree levels. The proposed features are thereby organized into five groups depending on the nature of the information extracted from the SOO tree, namely, tree shape, tree fitness, global fitness deviation, derivative and Lipschitz information, and tree dynamics. Although our primary goal is not to compete against the existing ELA features, but rather to complement them, we consider to study the relative effectiveness of the newly SOO-based features. Therefore, we report our investigations in solving a number of machine learning tasks in an attempt to assess in a systematic and automated manner the accuracy of the proposed features, when both compared separately to the existing ELA features, and combined with them. Getting its inspiration from the latest ELA developments, our assessment compromises standard tasks for (i) predicting the high-level features of the BBOB test suite, and (ii) the challenging task of selecting an accurate algorithm from a given portfolio. More specifically, our empirical analysis is based on the BBOB functions, and the data available from the COCO platform, for comparing continuous optimization algorithms. It is worth noticing that the BBOB test suite and the availability of a broad range of algorithms from the COCO platform contributed much in pushing the advent of the ELA methodology. Our comparative investigations is also to be viewed as a first step towards integrating the proposed features within more advanced ELA-based optimization and machine learning tasks.

*Outline.* In Section 2, a visualization of the SOO tree is given and the proposed features introduced. In Section 3, we study the relative importance of the proposed features when embedded in ELA classification and regression tasks. More precisely, in Subsection 3.2, we study the relative importance of the proposed features. In Subsection 3.3, we conduct a more advanced analysis on the relative accuracy of the proposed features when performing high level classification of the BBOB functions and when designing a simple algorithm selector. In Section 4, we conclude the paper.

**Algorithm 1:** Pseudocode of SOO

**Parameters:** $K$, and $h_{\max}$
$\mathcal{T} \leftarrow$ initial tree with one node representing the whole domain ;
**while** maximum number of evaluations not exhausted **do**
  $C \leftarrow \emptyset$; $v_{\min} \leftarrow +\infty$;
  **for** $h \in \{0, \ldots, \min(height(\mathcal{T}), h_{\max})\}$ **do**
    Among all leaves of $\mathcal{T}$ at height $h$, select the one, denoted $C_h^\star$, having the best fitness value at its center, denoted $x_h^\star$;
    **if** $f(x_h^\star) \leq v_{\min}$ **then** $C \leftarrow C \cup C_h^\star$; $v_{\min} \leftarrow f(x_h^\star)$ ;
  **for** $C \in C$ **do**
    split $C$ in $K$ sub-cells and add them accordingly in $\mathcal{T}$;

## 2 THE SOO-BASED FEATURES: FROM VISUAL INSPECTION TO FEATURE DEFINITIONS

For completeness, we provide a brief overview of the SOO algorithm and its theoretical foundations, as well as, a brief description of the BBOB functions used in the rest of the paper. The proposed features follow just after a visual inspection of the SOO behavior.

### 2.1 Background: SOO and BBOB in a Nutshell

*2.1.1 The SOO algorithm.* The so-called "Simultaneous Optimistic Optimization" (SOO) algorithm was originally introduced in [27]. It is a global optimizer finding its foundations in the machine learning field. It is a deterministic algorithm which has theoretically provable performance for functions being locally 'smooth' near their global optima, but where the actual smoothness is not known. This means that the function needs to not decrease too fast around at least one global optimum. More specifically, after $N$ function evaluations, the difference $f(x^*) - f^*$ between the value of the global optimum of the objective function $f^* = \min_{x \in X} f(x)$ and the value of the best point $x^*$ returned by SOO is decreasing in $O(N^{-1/d})$, where $d$ is the near-optimality dimension of $f$ [27]. Besides, an exponential rate $e^{-O(N)}$ can be achieved in the non-trivial case $d = 0$. These assumptions are much weaker than the function being globally Lipschitz, even with an unknown Lipschitz constant. In fact, only the performance of the algorithm is expressed as a function of smoothness (in a specific sense), but the algorithm design does not rely on this knowledge in any way. Interestingly, SOO is to our knowledge the only existing global optimizer for which the discrepancy between the best found point after a given number of function evaluations and the optimum can be bounded analytically under such a very weak assumption. That said from the theoretical side, the full pseudo-code of SOO is shown in Algorithm 1.

Despite its strong theoretical background, one can see that SOO is a very simple algorithm to set up in practice. It is a divide-and-conquer tree search algorithm, that share some common aspects with the so-called DiRect (Dividing Rectangle) algorithm [16]. It proceeds in an iterative manner by dynamically expanding a tree ($\mathcal{T}$) whose nodes are mapped to cells, also called hyper-rectangles, representing decreasing sub-domains. The initial root node represents the entire domain of the objective function. The algorithm considers the node at each height of the tree, i.e., nodes at distance $h$ from the root. Among the leaves at this height, it then selects the cell where the fitness value of its center has the lowest (in the case of minimization, largest in the case of maximization) fitness value $f(x_h^\star)$ (breaking ties arbitrary). The selected cell ($C_h^\star$) at height $h$ is split iff its value $f(x_h^\star)$ is lower than the value of all previously selected cells of lower heights — the value of a cell being defined as the value of the objective function $f$ at the center point $x$ of the sub-domain represented by that cell. Consequently, the selected cells are split and added in the tree after all the heights of the current tree were processed. This constitute one iteration of the algorithm. This process is then repeated until the budget of function evaluations is exhausted. Let us notice that SOO has only three simple parameters: $K$ the number of sub-cell to produce when performing splitting which is commonly set to 3, $h_{\max}$ the maximum size allowed when evolving the tree which usually set to a default value of $\Theta(\sqrt{\log^3(n)})$[1], and finally the direction in which cells are split which is usually done in a round-robin manner for each cell. One can find a first step towards the experimental evaluation of SOO in [32] and later in [10] using the well-established BBOB functions, which is also used in the reminder of the paper as our case study.

*2.1.2 The BBOB functions.* The continuous BBOB (**B**lack-**B**ox **O**ptimization **B**enchmark) testbed [13] include a diverse number of functions, allowing to study the relative behavior of optimization algorithms and to challenge them when tackling problems having different characteristics. Twenty four single-objective noiseless unconstrained functions are available, with a number of different instances available for each function. The BBOB functions distribute over a relatively broad range of properties, as summarized in Table 1. The test suite has 5 groups or *function types*, each one comprising a number of functions with variable properties in terms of *separability*, *multi-modality*, *global structure*, *variable scaling*, *search space* and *basin size homogeneity*, and *global to local optima contrast*.

### 2.2 A Visual Inspection

*2.2.1 Visualization of the SOO sample.* In Fig. 1, we show the solutions evaluated during the SOO search procedure for a some selected BBOB functions using their first available instance in dimension $d = 2$, i.e., two variables corresponding to the x- and y-axis, and a budget of $1000 \cdot d$. The different columns of the figure corresponds to the five function types of the BBOB tesbed, i.e., for each group, we visualize three selected functions. Let us also recall that it is commonly admitted that the function groups have different difficulty in terms of the number of function evaluations needed to approach the optimum up to some target value — functions being on the left side of the figure are commonly admitted to be relatively easier to solve than the ones being on the right side.

Different observations can be extracted from Fig. 1. First, we can see that SOO is able to sample from the whole variable space in a global manner, filling almost all regions, but in an unbalanced manner. Second, the point size and the color intensity, which are set in the figure relative to the iteration where the point was sampled and its fitness value respectively, inform about the properties of the explored points and SOO's behavior. For instance, one can see

[1] Notice that the maximum height value can only enter into play when a high number of function evaluations is used in order to prevent going unnecessary deep in the tree, i.e., the tree will naturally not exceed such a threshold in a practical scenario.

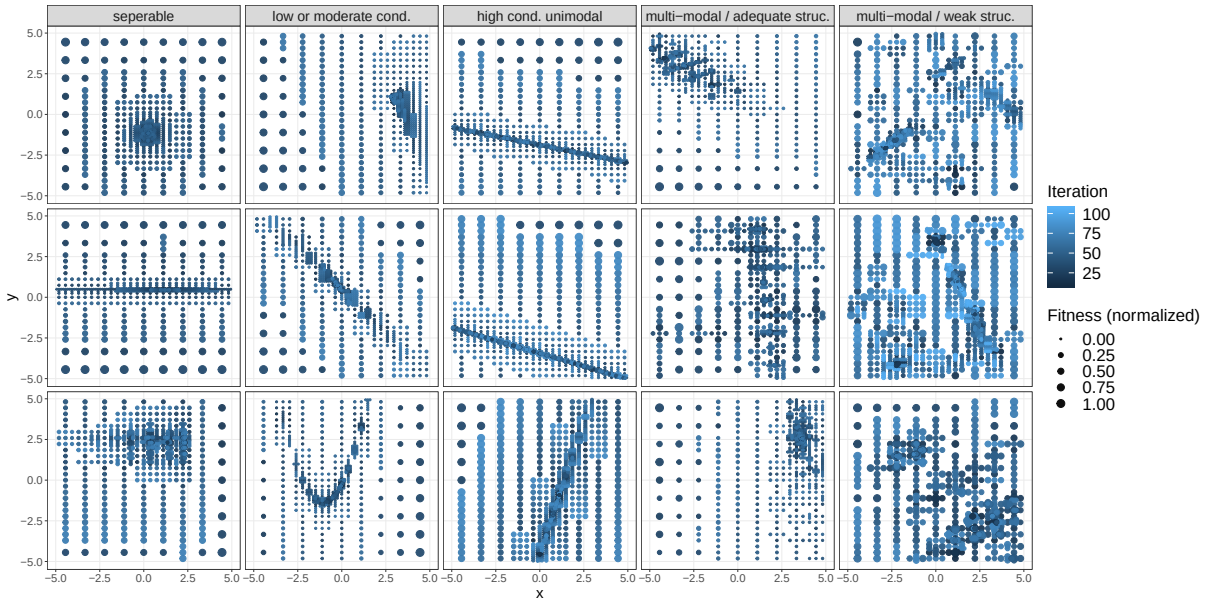


**Figure 1: Example of SOO sample points in $d = 2$ (respectively the x- and y- axis) using $1000 \cdot d$ function evaluations for some selected BBOB functions. From left to right, functions in each column belong respectively to the five BBOB groups, namely, Separable $(f_1, f_2, f_4)$, Low or moderate conditioning $(f_6, f_7, f_8)$, High conditioning and uni-modal $(f_{10}, f_{11}, f_{13})$, Multi-modal with adequate global structure $(f_{15}, f_{16}, f_{17})$, and Multi-modal with weak global structure $(f_{21}, f_{22}, f_{24})$.**

**Table 1: The 24 BBOB functions organized into 5 groups.**

| Function | multim. | gl.-struc. | separ. | scaling | homog. | basins | gl.-loc. |
|---|---|---|---|---|---|---|---|
| 1 Sphere | none | none | high | none | high | none | none |
| 2 Ellipsoidal seperable | none | none | high | high | high | none | none |
| 3 Rastrigin separable | high | strong | non | low | high | low | low |
| 4 Bueche-Rastrigin | high | string | high | low | high | med. | low |
| 5 Linear Slope | none | none | high | none | high | none | none |
| 6 Attractive Sector | none | none | high | low | med. | none | none |
| 7 Step Ellipsoide | none | none | high | low | high | none | none |
| 8 Rosenbrock | low | none | none | none | med. | low | low |
| 9 Rosenbrock rotated | low | none | none | none | med. | low | low |
| 10 Ellipsoidal high-cond. | none | none | none | high | high | none | none |
| 11 Discuss | none | none | none | high | high | none | none |
| 12 Bent Cigar | none | none | none | high | high | none | none |
| 13 Sharpe Ridge | none | none | none | low | med. | none | none |
| 14 Different Powers | none | none | none | low | med. | none | none |
| 15 Rastrigin mult-modal | high | strong | none | low | high | low | low |
| 16 Weierstrass | high | med. | none | med. | high | med. | low |
| 17 Schaffer F7 | high | med. | none | low | med. | med. | high |
| 18 Schaffer F7 ill-cond. | high | med. | none | high | med. | med. | high |
| 19 Griewank-Rosenbrock | high | strong | none | none | high | low | low |
| 20 Schwefel | med. | dec. | none | none | high | low | low |
| 21 Gallagher 101 Peaks | med. | none | none | med. | high | med. | low |
| 22 Gallagher 21 Peaks | low | none | none | med. | high | med. | med. |
| 23 Katsuura | high | none | none | none | high | low | low |
| 24 Lunacek bi-Rastrigin | high | weak | none | low | high | low | low |

that, depending on the degree of function difficultly, the correlation between the iteration when a point is sampled, and its fitness value can vary substantially. Besides, it appears that the maximum number of iterations that the algorithm is able to perform is also varying substantially depending on the tackled instance. More interestingly, we see that the distribution of the sampled points is different across functions, and has a tendency to draw some distinguishable patterns. Actually, a close look at the BBOB functions and their contour lines as provided in the comprehensive BBOB documentation, reveals that these patterns are directly related to the global shape of the functions. We can for instance easily recognize the sphere shape, as well as the weak structure of functions in the fifth group contrasting very clearly with the other groups.

These observations comfort us toward the benefits of using both the SOO dynamics and the underlying sample to extract informative landscape knowledge. Nevertheless, the previous discussion was only possible because a two dimension setting is considered. One may think that the SOO benefits may vanish when considering higher dimensions where it is difficult to have such visual observations. In the next section, we rather focus on the tree maintained by SOO as an alternative way to deal with this issue.

*2.2.2 Visualization of the SOO tree.* We use the well-known igraph R library in order to visualize the tree $\mathcal{T}$ constructed by the SOO algorithm. For the purpose of our study, we simply use the default rooted tree layout. The root of the tree represents the whole search space, and each intermediate node represents a cell before being split. In other words, the children of a node in the tree represents the sub-cells obtained after the cell of their parent node is split. The visualization of the tree obtained for the function considered previously in Fig. 1 is now shown in the same order in Fig. 2, but for dimension $d = 3$. An unzoomed example for the Ellipsoidal $f_{10}$ BBOB function for dimension $d = 2$ is further shown in Fig. 3 for the purpose of the discussion.

Fig. 3 shows that the SOO tree is not random, and has very specific characteristics. We can for instance see that the tree has a number of grapes while showing some properties in terms of node

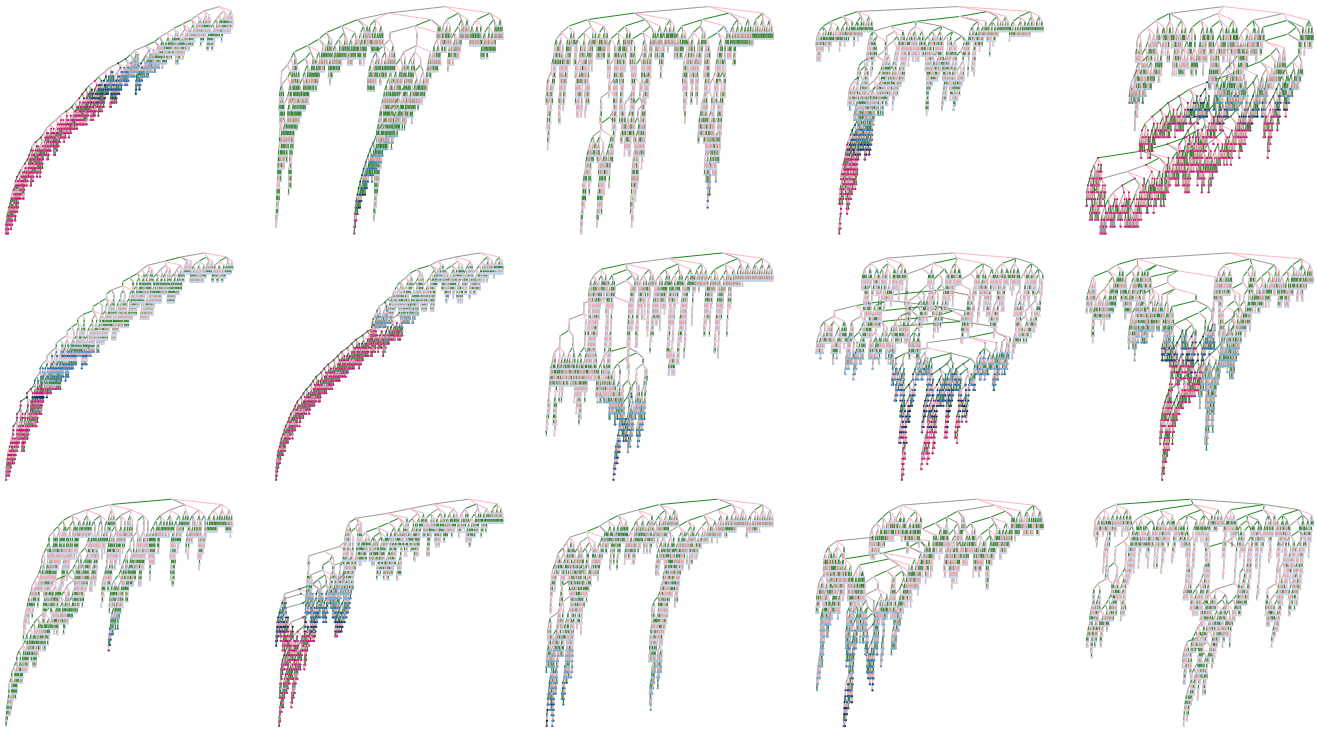

**Figure 2: Visualization of the SOO trees ($1000 \cdot d$ function evaluations). The tree in each cell is w.r.t the same function than in Fig. 1 but for $d = 3$.**

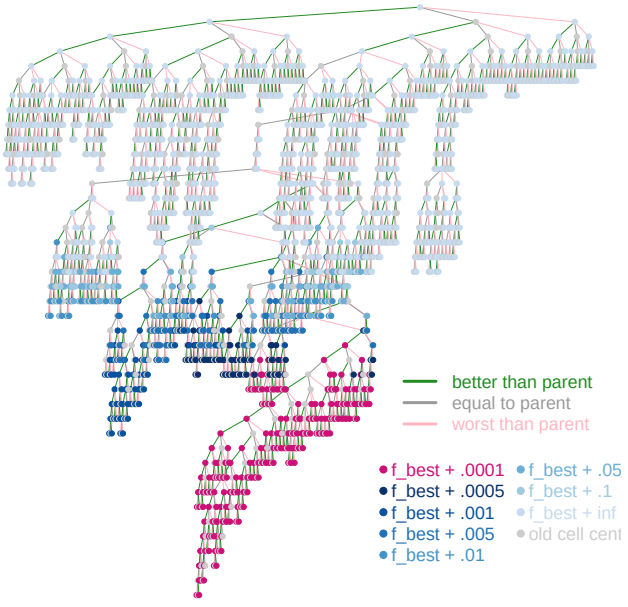


**Figure 3: The SOO tree for the $f_{10}$ BBOB function, $d = 2$ and $1000 \cdot d$ function evaluations.**

balance, as well as sub-trees and levels sizes. Moreover, we manage to visualize the edges of the tree in different colors depending on whether the center mapping a children cell is improving its parent or not (respectively, green and pink colors). We also attribute different colors to the nodes depending on the difference of the fitness values of their cell centers with the best point sampled by SOO. Notice that the center of a cell is evaluated only once although it appears several time in the tree, which is indicated in gray. Such encoding of edges and fitness values reveals that the tree contains some hidden information that informs about the landscape underlying the problem instance. At this point of the discussion, one should keep in mind that this reasoning could be misleading if SOO was not able to span diversified points of the variable space, which is actually the case as shown in Fig. 1. From Fig. 2, we can further observe that the shape of the SOO tree can be substantially different across the BBOB functions. We can see the simple shape of the deep imbalanced tree in the top left corner (sphere), whereas we notice more complex tree shapes for the most difficult functions, exposing various grapes and long paths likely w.r.t. multiple optima.

Although this visual inspection is informative, we need to translate it in some numerical values so that we can achieve our main goal which is the design of landscape features that can be integrated within an ELA approach.

## 2.3 SOO Features Definition

In the following, we propose a set of features in an attempt to capture the shape of the constructed tree and the information encoded by the underlying fitness values of nodes. It is worth noticing that a large specialized literature on the topological structure of trees, as well as on studying specific tree distances and tree metric spaces, is available from graph theory, pattern recognition, machine learning, bio-informatics, and many other connected fields, which is with no surprise since trees are fundamental structures that appear in a number of applied and theoretical fields. To keep our features as simple as possible and their computation reasonable and scalable in practice, we focus on relatively simple statistics. The designed features are hence obtained as a particular aggregation of well-defined node and edge attributes. In the following, having a finite set of (ordered) real valued numbers (attributes) $\mathcal{N} = (n_0, n_1, \ldots, n_i, \ldots)$, we shall use besides their average (avg) value $\overline{n}$ and their standard deviation (sd), the so-called first and second auto-correlation coefficients $r_1$ and $r_2$, defined as follows: $r_k = \sum_{i=1}^{|\mathcal{N}|-k}(n_i - \overline{n}) \cdot (n_{i+k} - \overline{n}) / \sum_{i=1}^{|\mathcal{N}|}(n_i - \overline{n})^2$.

Let us recall that each tree node $u$ represents a cell in the variable space. In one iteration of SOO, that is in one execution of the outer while loop, a number of nodes can be selected for splitting. If a node $u$, called parent, is split into $K$ sub-cells, the tree is expanded and $K$ new nodes $(u_1, \ldots, u_K)$, called children, are added. A node is termed leaf if it has no child (i.e., it has not been selected for splitting). Node $u$ is called an ancestor of node $v$, if $u$ is the parent of $v$, or $u$ is the ancestor of $v$'s parent. For any integer $r \in \{1, 2, \ldots, K\}$, a set of $r$ leaves having a common parent, is termed an $r$-cherry. A node that is not a leaf is termed an internal node. The sub-tree of a node $u$, denoted $\mathcal{T}_u$, is defined as the tree rooted at $u$ and containing all nodes obtained after splitting $u$ and its subsequent children. We say that $u$ is the ancestor of $v$, if $v$ belongs to $\mathcal{T}_u$. Finally, we view the tree as organized into $h + 1$ ordered levels, with $h$ being the height of the tree. For each $\ell \in \{0, \ldots, h\}$, a tree level is denoted $\mathcal{L}_\ell$ and contains by definition all nodes $u$ at height $h_u = \ell$.

The set of proposed features are organized into five groups and summarized in Table 2. In the first four groups, we consider the final tree obtained after the SOO algorithm terminated. The first group of features, called *Tree Shape* features, deals with the shape of the constructed tree without any consideration about the fitness values of evaluated points. The second group, called *Tree Fitness* features, is an attempt to leverage the features of the first group in order capture the fitness distribution among the nodes of the tree. The third group, called *Global Deviation* features, is based on the deviation of fitness values observed among the nodes in the different sub-trees. The fourth group, called *Derivative and Lipschitz*, is an attempt to provide an estimation of the function derivative and the Lipschitz coefficient based on the fitness of sub-trees rooted at each level. Finally, the last group, called *Tree Dynamics*, provides features informing about the evolution of the tree and the dynamics of its construction during the different iterations of SOO.

*2.3.1 Tree shape (TS).* Let $h_v$ denote the height of node $v$, that is the length of the path leading to its root. Let the age of node $v$, denoted $a_v$, be the length of the longest shortest path in the tree leading to a leaf. Let us denote by $s_\ell$ and $t_\ell$ respectively the number of nodes and the number of leaves at level $\mathcal{L}_\ell$.

The first group of proposed features is presented in Table 2 for a better clarity. These features inform about the shape of the tree. They range from very simple ones, such as the height, the size, the number of leaves, etc., to more sophisticated ones such as statistics about the age of nodes, the size or the leave distribution among the levels of the tree. They also include a number of standard statistics from the literature such as the sackin index [35], which is one of the oldest statistics to summarize the shape of a tree, an adaptation of the Colless tree imbalance statistic [8] commonly used for binary trees, and the cherries statistics [23] used in phylogenetic tree analysis, e.g., [6]. Notice that a number of designed features relate to statistics about the tree levels. This is motivated from two perspectives. First, from our visual inspection, we can see that the distribution of nodes in the different levels can differ substantially from a function to another, hence deeply impacting the tree shape. Second, by the design of SOO itself, the shape of the tree is conditioned by the iteration when the leaves of each level are selected, i.e., the inner for-loop. This is clearly related to the fitness landscape encountered when sampling from the different levels of the tree. To further capture this aspect, the next group of features attempts to render the fitness values across the nodes.

*2.3.2 Tree fitness (TF).* Let $x_v$ denote the center of the cell corresponding to node $v$. Let $f_v$ denote the fitness of node $v$ defined as $f_v = f(x_v)$. Let $\omega_v$ denote the weight of node $v$, defined as $\omega_v = \sum_{u \in T_v} f_u$. Let the weight $\omega_{(u,v)}$ of an edge $(u, v)$ (connecting the parent node $u$ to the child $v$) be the difference between the fitness of $u$ and the fitness of $v$, i.e., $\omega_{(u,v)} = f_u - f_v$. Similarly, let the reward $r_{(u,v)}$ of an edge $(u, v)$ be 0, 1, or $-1$ when $\omega_{(u,v)}$ is respectively equal, greater, or less than 0. Notice that an edge with a strictly positive fitness weight (or equivalently, a reward equal to 1) means that the child is improving its parent (for minimization).

With these definitions, the second group of features can again be found in Table 2. In particular, we focus on capturing how smooth and uniform is the fitness distribution over the different levels of the tree, e.g., the `TF_xLf_*` features. For example, when considering a simple sphere function, one can expect that the average fitness of nodes is correlated over the tree levels, since as the tree goes deeper, the fitness is likely to improve towards the global optimum. In contrast, for a very rugged and unstructured landscape, the fitness among the different levels is likely to be uncorrelated. Besides, from a theoretical perspective, the SOO algorithms makes use of the best fitness values of each level as a sort of upper bound when the exact near-optimality dimension is not known, and where smoothness near the global optimum being a sort of a locally one-sided Lipschitz assumption [27]. Roughly speaking, this group can be viewed as a way to capture a degree of smoothness of the landscape, which is actually complemented by the features from the next two groups.

*2.3.3 Global deviation (GD).* Let $L_u$ be the set of nodes $v$ in the sub-tree $\mathcal{T}_u$ (rooted at $u$) such that $v$ is a leaf and $f_v \neq f_u$. For every node $v \in L_u$, we define its fitness deviation w.r.t $u$ as follows: $d_v^u = \frac{f_u - f_v}{|f_u|}$. For every node $u$, we consider the maximum, the average and the standard deviation, to which we refer using respectively letters `m`, `a`, and `s`, of the $d_v^u$ values over the nodes $v \in L_u$. For each node $u$, we also consider the number of its improving leaves (referred to with letter `ni`), i.e., the number of nodes $v \in L_u$ s.t. $d_v^u >$

**Table 2: The proposed 214 tree-based features organized in five groups: Tree Shape (37 TS features), Tree Fitness (39 TF features), Global Deviation (40 GD features), Derivative and Lipschitz (80 DL features), and Tree Dynamics (18 TD features).**

| Feature name | Definition |
|---|---|
| TS_s, TS_l | Tree size and number of tree leaves. |
| TS_h_max, TS_h_avg, TS_h_sd | Maximum, average and standard deviation of the heights of nodes |
| TS_lh_avg, TS_lh_sd | Average and standard deviation of the heights of leaves |
| TS_sackin | The sackin index [35] of the tree normalized by the number of nodes TS_s: Sum over all leaves of the number of internal nodes (including the root) between each leaf and the root, i.e., $\sum_{v \text{ is leaf}} h_v$ |
| TS_cherries_$r$, TS_ncherries_$r$ (where $r \in \{1, 2, 3\}$) | Number and normalized number of $r$-cherries [23]. Normalization is w.r.t. TS_s, i.e., TS_ncherries_$r$ = $\frac{\texttt{TS\_cherries\_}r}{\texttt{TS\_s}}$ |
| TS_colless, TS_colless_avg, TS_colless_sd | Colless-like statistics [8]: Sum, average and standard deviation over all internal nodes of the maximum pairwise differences of the sub-tree sizes of children |
| TS_subt_avg, TS_subt_sd | Average and standard deviation of the sub-tree sizes $(\lvert\mathcal{T}_{v_0}\rvert, \ldots, \lvert\mathcal{T}_{v_i}\rvert, \ldots)$ of tree nodes $(v_0, \ldots, v_i, \ldots)$ |
| TS_L_avg, TS_L_sd, TS_nL_avg, TS_nL_sd | Average and standard deviation of the sizes $(s_0, \ldots, s_\ell, \ldots)$, respectively the normalized sizes $(s_0/0^K, s_1/1^K, \ldots, s_\ell/\ell^K, \ldots)$, of the tree levels |
| TS_lL_avg, TS_lL_sd, TS_lL_r1, TS_lL_r2 | Average, standard deviation, first and second autocorrelation coefficient of the number of leaves $(t_0, \ldots, t_\ell, \ldots)$ at each tree level |
| TS_age_avg, TS_age_sd | Average and standard deviation of the ages $(a_{v_0}, \ldots, a_{v_i}, \ldots)$ of the tree nodes $(v_0, \ldots, v_i, \ldots)$ |
| TS_aLage_avg, TS_aLage_sd, TS_aLage_r1, TS_aLage_r2 | Average, standard deviation, first and second autocorrelation coefficient of the average age of nodes at each tree level |
| TS_sLage_avg, TS_sLage_sd, TS_sLage_r1, TS_sLage_r2 | Average, the standard deviation, the first and the second autocorrelation coefficient of the standard deviation of the ages of nodes at each tree level |
| TF_hb, TF_nhb | Minimum height and normalized minimum height of the node with the best fitness. |
| TF_fhcorr_$x$ (where $x \in$ {prs, spr, ken, cov}) | The pearson coeff. ($x$ = prs) (respectively, spearman ($x$ = spr), kendal ($x$ = ken) and covariance ($x$ = cov)) between nodes' heights ($h_v$ values) and fitnesses ($f_v$ values) |
| TF_$x$Lf_avg, TF_$x$Lf_sd, TF_$x$Lf_r1, TF_$x$Lf_r2 (where $x \in$ {b, a, s}) | Average, standard deviation, first and second autocorrelation coefficient of the $x$ statistics of the tree levels. For $x$ = b (respectively $x$ = a and $x$ = s), the best (respectively. average and standard deviation) fitness over the nodes of each level is considered |
| TF_nodew_avg, TF_nodew_sd | Average and standard deviation of nodes' weights $(\omega_{v_0}, \ldots, \omega_{v_i}, \ldots)$ |
| TF_colless, TF_colless_avg, TF_colless_sd | Sum, average and standard deviation of the maximum pairwise differences of the children weights over all internal nodes |
| TF_edgew, TF_edgepw | Sum of edge weights, i.e., $\sum \omega_{(u,v)}$ and sum of (strictly) positive edge weights, i.e., $\sum \max\{0, \omega_{(u,v)}\}$ |
| TF_edger, TF_edgepr | Sum of edge rewards and sum of (strictly) positive edge rewards (i.e., number of nodes improving their parents) |
| TF_ba, TF_lba | Number of nodes and number of leaves that are better than their ancestors, i.e., all edges on the path leading to the root have 0 or 1 reward |
| TF_bc_avg, TF_bc_sd | Average and standard deviation (over the nodes) of the number of children connected with edges having a reward of 1 |
| TF_$x$Lbc_avg, TF_$x$Lbc_sd, TF_$x$Lbc_r1, TF_$x$Lbc_r2 (where $x \in$ {a, s}) | Average, standard deviation, first and second autocorrelation coefficient of the $x$ statistics of the tree levels. For $x$ = a (respectively $x$ = s), the average (respectively, standard deviation) of the number of children connected with edges having a 1 reward, over the nodes in each level, is considered |
| GD_$xy$_avg, GD_$xy$_sd, GD_$xy$_r1, GD_$xy$_r2 (where $x \in$ {a, s} and $y \in$ {m, a, s, ni, pi}) | Features based on fitness deviation. See subsection 2.3.3 for the detailed definitions. |
| DL_$xyz$_avg, DL_$xyz$_sd, DL_$xyz$_r1, DL_$xyz$_r2 (where $x \in \{a, s\}$, $y \in \{a, s\}$ and $z \in \{d, l2, l4, l8, linf\}$) | Features based on the estimation of the derivative and the Lipschitz coefficient. See subsection 2.3.4 for the detailed definitions. |
| TD_iters | Number of iterations performed by SOO. |
| TD_split_avg, TD_split_sd, TD_split_r1, TD_split_r2 | Average, standard deviation, first and second autocorrelation coefficient of the number of splits $(s_0, s_1, \ldots, s_i, \ldots)$ performed over SOO iterations. |
| TD_nsplit_avg, TD_nsplit_sd, TD_nsplit_r1, TD_nsplit_r2 | Average, standard deviation, first and second autocorrelation coefficient of the normalized number of splits $(s_0/h_0, s_1/h_1, \ldots, s_i/h_i, \ldots)$ performed over SOO iterations. |
| TD_h_avg, TD_h_sd, TD_h_r1, TD_h_r2 | Average, standard deviation, first and second autocorrelation coefficients of the tree heights $(h_0, h_1, \ldots, h_i, \ldots)$ over the different iterations. |
| TD_iterh_neutral | Number of successive iterations where the height of the tree remains unchanged (neutral), i.e., number of $i$ s.t. $h_i = h_{i+1}$. |
| TD_iterh_diff, TD_niterh_diff | Number of iterations where the tree height $h_i$ increased, and its normalized variant TD_niterh_diff = TD_iterh_diff/TD_iters. |
| TD_iterh_avg, TD_iterh_sd | Average and standard deviation of the iteration number $i$ where the height $h_i$ of the tree increased, i.e., at which iterations the tree height increased. |

0, and the proportion of improving leaves in $L_u$ (referred to with letter pi), i.e., pi = ni/$|L_u|$. For every level, we consider the average and the standard deviation statistics computed over the nodes of that level. Hence, we obtain $5 \times 2$ statistics for each level. Now, we consider the average, the standard deviation, the first and the second autocorrelation coefficients of those statistics computed over the different levels, hence ending up with $5 \times 2 \times 4 = 40$ features that we denote GD_xy_w where $x \in \{a, s\}$, $y \in \{m, a, s, ni, pi\}$ and $w \in \{avg, sd, r1, r2\}$, e.g., the feature GD_as_avg reads as the average (over the levels) of the average (over the nodes $u$ in each level) of the standard deviation (over nodes $v$ in $L_u$) of the fitness deviation.

The obtained features can be motivated as follows. In the initial design of SOO, the fitness of the center of a cell is thought as providing an estimation of how promising is the quality of the cell, and how likely it is to lead to an optimum. Hence, choosing the best cell in each level intensifies the search around promising regions, whereas choosing eventually a cell from *each* level ensures some exploration degree. The exploitation-exploration trade-off is further controlled by the rule that a smaller cell being deeper in the tree, is not expanded if its quality is lower than a larger cell being higher in the tree. In the case a node is in a smooth and well-shaped 'basin of attraction' of an optimum, it is likely that its sub-tree gets expanded and the fitness deviation of the nodes in its sub-tree improved. On the contrary, for non-smooth unstructured functions, it is likely that the fitness deviation follows a completely different trend. Besides, the probability that a node gets improved when splitting its cell is likely to be a function of the landscape around its center and the size of its cell (its height). The designed features are an attempt to render such an intuition by measuring the fitness deviations across multiple cells.

*2.3.4 Derivative and Lipschitz coefficient estimation (DL).* This group of features aims at providing an estimation of the distribution of the derivative of the function and a Lipschitz-like coefficient computed at different regions (cells) of the landscape sampled by SOO, which is known to have a deep impact impact on algorithm design and search behavior. For every node $v \in L_u$, we define the derivative-like coefficient $\frac{f_u - f_v}{x_u - x_v}$ and the Lipschitz-like coefficient $\frac{|f_u - f_v|}{||x_u - x_v||}$ with respect to $u$. For every node $u$, we similarly consider the average and the standard deviation statistics with respect to these coefficients, computed over the nodes of $L_u$. For every level $\ell$ in the tree, we again consider the average and the standard deviation statistics computed over the nodes belonging to the considered level. Finally, we consider the average, the standard deviation, the first and the second autocorrelation coefficients of these statistics computed over the different levels of the tree. However, we consider four different norms ($||\cdot||$) when computing the Lipschitz like feature, namely, the $p$-norm with $p \in \{2, 4, 8, \infty\}$. Hence, we end up with $(1 + 4) \times 2 \times 2 \times 4 = 80$ features, that we denote DL_xyz_w where $x, y \in \{a, s\}^2$, $z \in \{d, l2, l4, l8, linf\}$ and $w \in \{avg, sd, r1, r2\}$, e.g., the feature DL_asl2_r1 reads as the first autocorrelation coefficient (over the levels) of the averages (over the nodes $u$ in each level) of the standard deviation (over nodes $v$ in $L_u$) of the Lipschitz estimation computed using the 2-norm.

*2.3.5 Tree dynamics (TD).* The last group of features is w.r.t the dynamics of the tree over the different iterations which is to contrast with the four previous groups. Let $h_i$ be the height of the tree at iteration $i$. We define the number of splits $s_i$ performed at iteration $i$ as the number of nodes that are selected for expansion, i.e., the size of set $C$ in the inner for-loop of Algorithm 1. Notice that SOO can split at most one node at each level of the tree, hence being able to split at most $h_i$ nodes in a given iteration $i$. The definitions of our last proposed features are then provided in Table 2.

All features in this group relates to how the tree is expanded. For instance, a higher number of iterations, when comparing two landscapes, means that less nodes are split at each iterations (features TD_iters and TD_nsplits_avg). This means that nodes being deeper in the tree have more often worst fitness values than nodes being higher. This can typically indicates that SOO is missleaded when splitting cells, or has found a new 'basin of attraction'. This can indicate that two landscapes have different distributions of local optima or have seemingly different structures. Similarly, the speed at which the height of the tree increases can provide information about the relative quality of local optima, and how easy it is to jump in the landscape to find a new high-quality solution.

## 3 EXPLORATORY LANDSCAPE ANALYSIS

In this section, we study the SOO features by adopting an Exploratory Landscape Analysis (ELA) approach [19, 24]. In fact, on the basis of the extracted low-level landscape features, two main general tasks have been considered so far, namely, *high-level property classification* and *algorithm selection*. These two tasks are still open and are being tackled using a variety of machine learning techniques. Our goal is not to close them, but to show how the designed features can be embedded in an ELA approach and to push a step forward its development. Hence, we first briefly recall some ELA background needed to understand the adopted methodology.

### 3.1 Preliminaries and Set Up

*3.1.1 The high-level properties classification task.* The first task we shall consider in our analysis consists in predicting expert and high-level properties of an optimization problem, as those used to form the different groups of the BBOB functions (See Table 1). This is motivated by the fact that such high-level properties could not be available for an unseen blackbox optimization problem, and by the assumption that knowing them for a particular optimization problem could help deciding on an accurate and efficient algorithm.

*3.1.2 The algorithm selection task.* The second task consists in predicting the performance of one or a set of solvers, which is connected to the Algorithm Selection Problem (ASP) [18]. Generally speaking, assuming a portfolio of algorithms, and given an unknown (unseen) problem instance, one aims at selecting the algorithm that is believed to perform best. One or several models are built during a training phase, which are then used for the sake of predicting the best performing algorithm in the testing phase. The target models are trained using the landscape features extracted from the training instances, and an information about the (relative) algorithm performance. In the case of the BBOB testbed, a fine-grained data about the performance of a wide range of algorithms

is available [13] as a byproduct of the last ten years benchmarking efforts. In fact, we consider the data available from the COCO platform [29] for **CO**mparing **C**ontinuous **O**ptimization algorithms.

*3.1.3 Solvers portfolio and ERT specification.* Following the latest ELA developments [19], we consider a portfolio of 10 algorithms among more than a hundred solvers available from COCO, namely,

- A deterministic algorithm (**Brent-STEPrr**) [31] which is a variant of the Brent-STEP algorithm performing axis-parallel searches with a round robin strategy.
- Five multi-level solvers [30], namely, the multi-level single linkage method (**MLSL**), an interior-point version for constrained non-linear problems (**fmincon**), a quasi-Newton version (**fminunc**) approximating the Hessian using the BFGS method, a hybrid variant improving the sampling phase (**HMLSL**), and a multi-level coordinate search (**MCS**) [15].
- Three solvers based on Covariance Matrix Adaption Evolution Strategy (CMA-ES), namely, CMAE-ES with cumulative step-size adaptation (**CMA-CSA**) [1], a restart version of the CMA-ES with an increasing population size (**IPOP400D**) [2], a hybrid variant (**HCMA**) [3].
- A commercial multi-start algorithm (**OQNLP**) [30, 38].

None of these algorithms outperforms all the others on all functions. However, they show extremely efficient behavior on some functions or groups of functions, hence making them good candidates to be included in a portfolio [19]. As a measure of performance, we also follow the existing work [19], and we consider the so-called relative Expected Runtime (relERT) using the *five first* instances of each BBOB function and a target threshold of $\epsilon = 10^{-2}$ to the optimum. From the COCO data, and for a given function at dimension $d$, we can decide whether a solver is successful to approximate the global optimum of instance $i \in \{1, 2, 3, 4, 5\}$ up to a precision value $\epsilon$ (in the objective space) and how many function evaluations $FE_i(\epsilon)$ were performed until the solver terminated (successfully or not). The ERT [12] of the solver is defined as $ERT(\epsilon) = \frac{\sum_i FE_i(\epsilon)}{\sum_i success_i(\epsilon)}$ where $success_i(\epsilon) = 1$ if the solver was successful and 0 otherwise. The relERT of a solver is then defined as the normalization of the ERT using the ERT of the best (among the 10) solver for that function at the considered dimension[2], i.e., a value $x$ means that the ERT of the considered algorithm is $x$ times slower than the algorithm with the best ERT on that particular function.

*3.1.4 Landscape features set up.* In addition to the proposed SOO features, we consider the set of features existing from the ELA literature, which are then termed as ELA features. For this purpose, we use the recently proposed flacco R package [17], providing a unified interface to a collection of feature sets including the latest developments made by the community. We consider to compute all available features from the following flacco feature sets: cell mapping angle (8 `cm_angle`), cell mapping convexity (4 `cm_conv`), cell mapping gradient homogeneity (2 `cm_grad`), distribution (3 `ela_distr`), level set (18 `ela_level`), meta model (9 `ela_meta`), general cell mapping (73 `gcm`), barrier tree (89 `bt`), information content (5 `ic`), basic (13 `basic`), dispersion (16 `disp`), nearest better clustering (5 `nbc`), principal component analysis (8 `pca`). The number of considered ELA features is then 199 overall, i.e., slightly less than the number of proposed SOO features (214).

All existing ELA features require an initial sample from the variable space and the corresponding objective values. The sample points are computed by a Latin Hypercube Sampling (LS), as recommended in the literature [19], with a budget of $s \times d$ function evaluations for each instance and feature, with $s \in \{50, 100, 1000\}$ and $d$ is the variable space dimension. The same budget ranges are also used to compute the proposed SOO features. Notice that for $s \in \{50, 100\}$, we target a cheap setting for feature computation and analysis, whereas for the remaining value $s = 1000$, we aim at looking at the impact of higher budgets. Besides, to keep the computational effort needed to conduct our experiments reasonable, we restrict the set of dimensions to $d \in \{2, 3, 5\}$, and cell mapping features are computed using a division of the variable space into 3 cells per dimension[3]. Notice also that the ELA features sets available from the flacco R package and requiring additional function evaluations were excluded, e.g., the local search based features, as it is usually the case in most recent ELA studies [19].

## 3.2 Feature Importance

The aim of this section is to highlight the relative relevance of the designed SOO features with respect to existing ELA features. Therefore, we rely on tree-based predictive models which allow to identify which input features are the most important to make accurate predictions [21]. More precisely, we use a random forest model [7] with a standard (default) setting of 500 trees and no further tuning. We thereby consider 8 (independent) classification tasks aiming at, respectively, learning the 7 high-level properties of the BBOB functions and at predicting their high-level group. We also consider 10 (independent) regression tasks aiming at predicting the relERT of the 10 algorithms from our portfolio. For every task, we build a random forest model per dimension and per budget ($18 \times 3 \times 3 = 162$ models), while mixing the SOO and ELA features. We then compute the importance of features provided by every model, which is taken to be the Mean Decrease in Accuracy in the case of a classification task, and the Node Impurity in the case of a regression task. For each model, we then rank the features according to the so-obtained importance values.

In Fig. 4, we show among the top $N$ most important features, and as a function of $N \in \{1, 2, 3, \ldots, 50\}$, the percentage of those being in the proposed SOO features and those being in the existing ELA features, averaged over the different budgets and the different underlying models. Two main observations can be extracted. First, we clearly see that, among the top 10 ones, the proportion of SOO features appears in a non-negligible ratio, indicating that they play an important role in the machine learning tasks. They appear even more often than the ELA features in all scenarios when examining the top 10 to 50 important features (among the 413 ones). Second, we clearly see that the the top $N$ important features contains always more SOO features than ELA features for the regression tasks, whereas it is not always the case for the classification tasks. This indicates that the SOO features are extremely relevant for

[2] In case a solver is unsuccessful in all instances, we impute the missing (infinite) value using a PAR10 score.

[3] Actually, this is the maximum that we could set to compute ELA features using an Apple Macbook Pro with a 2,6 GHz Intel Core i7 CPU and 8 Gb RAM.

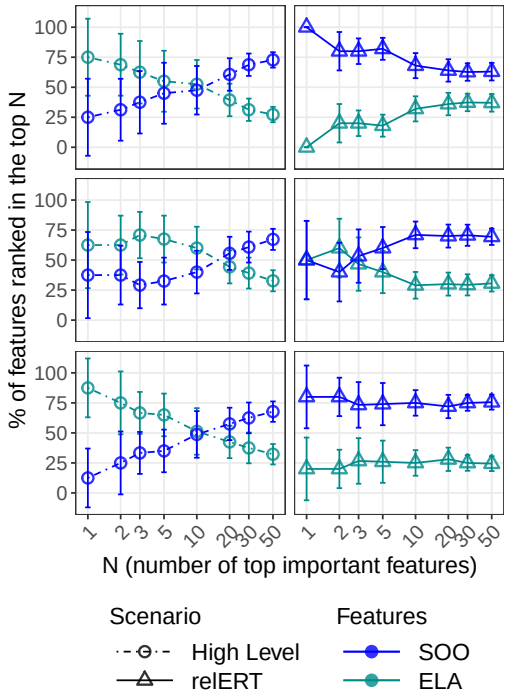


**Figure 4: Among the top $N$ important features, we show the (average) percentage of SOO and ELA features respectively to the classification (High-level properties, Left column) and the regression (relERTs, right column) tasks. The rows from top to bottom correspond to dimensions 2, 3, and 5.**

the prediction of algorithms relERT, while they are relatively less important than the ELA features for predicting the high-level properties. This can be explained by the fact that the SOO features are extracted from the landscape trace of a concrete optimization algorithm, while the ELA features are extracted from a simple Latin Hypercube Sampling design, independent of any optimization process. Hence, these first results indicate that the SOO features are complementary to the existing ELA features and are more likely to play an important role in predicting the difficulty encountered by an algorithm when searching a particular landscape.

In Fig. 5, we further provide a hint on the SOO features that are found the most important. More precisely, we compute all SOO features that are ranked in the top 3 important ones at least once over all experimented scenarios. Those features are then shown in the $x$-axis, and a colored mark is used to indicate the rank(s) of each feature with the corresponding task depicted in the $y$-axis. For instance, the `GD_api_avg` feature (See Table 2), belonging to the Global Deviation group, and referring to the average (over the tree levels $\mathcal{L}_\ell$) of the average (over the nodes $u$ in a ) of the proportion of improving leaves in the underlying sub-trees ($\mathcal{T}_u$), is found to be at least once the most important feature for predicting the multimodality high-level property, and for predicting the relERT with respect to **HCMA**, **MCS**, **fminunc**, and **BrentSTEPrr**. It is also found to be at least once in the top 2 most important features for the relERT prediction of **MLSL**, and in the top 3 most important features with respect to **fmincon** and **OQNLP**.

Besides providing an idea about the most important features, two observations can be extracted from Fig. 5. First, the five proposed groups are all well represented in the top 3 important features. The figure actually shows 86 features being at least once in the top 3 most important features, which represents 40% of the proposed features. Second, one can notice that the SOO features are present more often in the regression tasks (relERT prediction) than in the the high-level classification tasks, which confirms our previous finding. However, this appears to be dependent on the group of SOO features. Roughly speaking, we find that the Tree Shape, Tree Fitness, and Global Deviation groups are more often important for relERT prediction than for high-level properties prediction, whereas the reverse holds for the Derivative, Lipschitz and Tree Dynamic groups – with few exception e.g., the `DL_aad_avg` which appears at least once as the most important features for four out of the ten solvers. Of course, such observations are to be mitigated with the possible complex interaction between the different features, as well as the fact that we are only looking at the top 3 important features among more than 400 SOO and ELA features.

In Fig. 6, we show the boxplots summarizing the values of some features ranked among the top 3 important in the previous analysis. We can see that no single feature allows to discriminate in a clear manner between the different BBOB groups, which is with no surprise given the initial purpose of the BBOB functions. However, we can see some clear trend in the distribution of some features values, as the assumed difficulty of the function increases. For instance, the `TD_nsplit_avg` feature (last column), which is the average normalized number of splits over iterations, seems to correlate negatively with function difficulty, i.e., the number of splits is larger for the sphere and separable functions, and it clearly decreases as the characteristics of the function become more complex.

Additionally, Fig. 6 provides an interesting hint on the robustness of features w.r.t the budget used to extract them. In fact, it seems that the relative distribution of feature values across the BBOB functions is seemingly the same when considering different budgets. This observation provides confidence that a low budget of function evaluations can be sufficient to render a relevant information about the landscape. Of course, this claim is to be investigated further in the future in order to better elicit the impact of the budget on the accuracy of the proposed SOO features. In the rest of the paper, we shall focus of a cheap budget setting when computing both the SOO feature set and the ELA feature set, while assessing their relative performance in a systematic manner for more advanced tasks.

## 3.3 Features Analysis with Cross-validation

A crucial aspect in an ELA approach, as in any learning task, is the necessity to validate the constructed models on unseen (test) instances [5]. While the previous analysis provides already insights into the relevancy of the SOO features, the considered models were trained using the whole data set (features computed for all functions). Following previous ELA studies, we push our analysis further by cross-validating either on the BBOB instances or functions:

- *Leave-one-instance-out cross-validation (LOIO CV)*: this consists in leaving out one instance (out of the 5 considered ones) in the training phase for each BBOB function (hence having $24 \times 4$ instances), and validating the constructed model(s) on

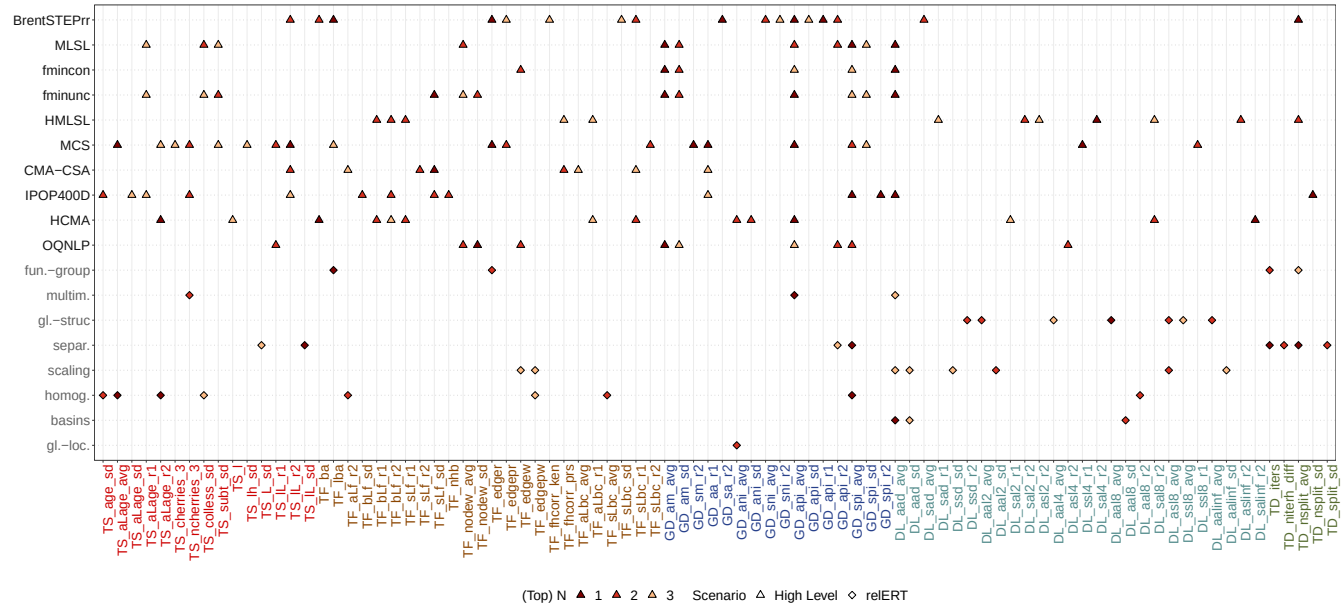


**Figure 5: SOO features appearing in the top 3 most important features at least once over all tasks and configurations, i.e., high-level properties or relERTs, dimensions and budgets.**

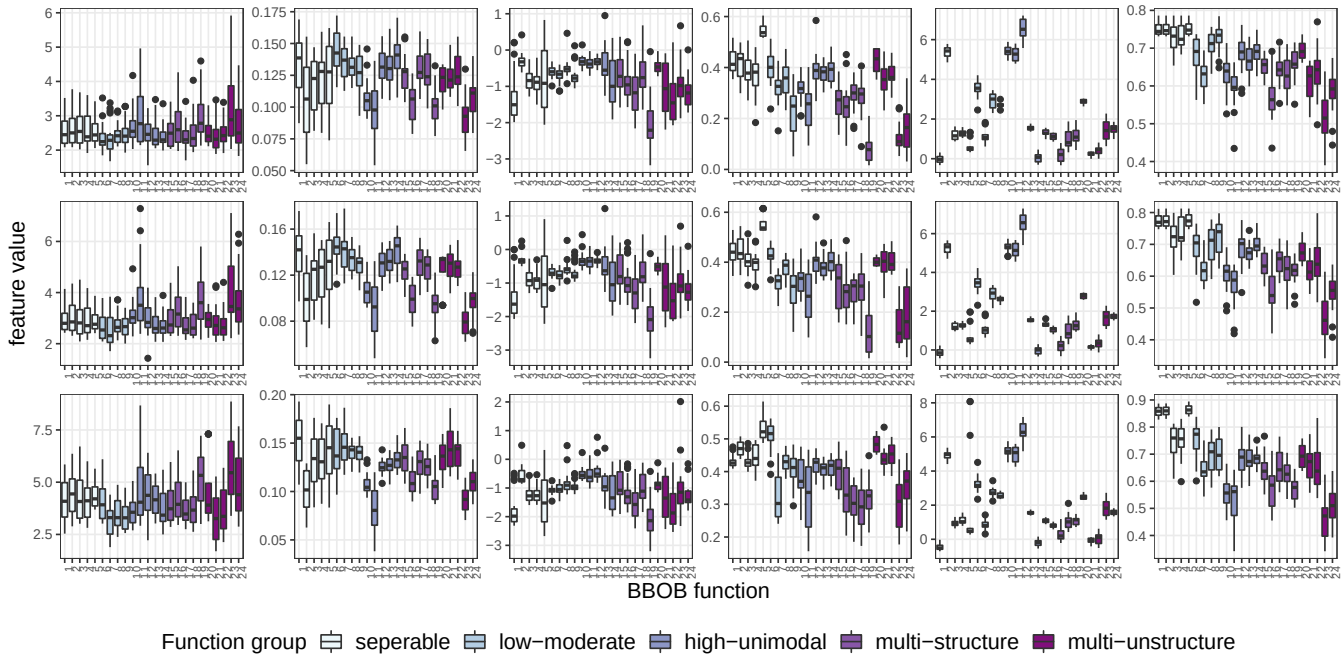


**Figure 6: Example of SOO feature values (y-axis) over the 3 dimensions and the 5 instances for the 24 BBOB functions (x-axis). Columns are with respect to different features, respectively from left to right: `TS_age_sd`, `TS_ncherries_3`, log(`GD_am_avg`), `GD_api_avg`, log(`DL_aal2_avg`) and `TD_nsplit_avg`. Rows are with respect to different budgets ($s \cdot d$) with $s$ being equal to, respectively from top to bottom, 50, 100, and 1000.**

the remaining 24 instances. In other words, we perform a 5-fold cross-validation where, in each sequence, 96 instances are used for training and 24 for testing.

- *Leave-one-problem-out cross-validation (LOPO CV)*: this consists in leaving out all the instances of a problem in the training phase, and to use them solely in the validation (testing) phase. In other words, we perform a 24-fold cross-validation, where, in each sequence, the 5 instances of 23 functions

(out of the 24 BBOB ones), are used for training, and the 5 instances of the remaining function are used for testing.

A LOPO CV can be motivated by studying how well an ELA approach is able, not only to generalize over unseen instances, but also to transfer knowledge to unseen functions. However, knowing that the BBOB test suite was specifically designed to provide a diverse set of functions, the feature space of the discarded functions in a LOPO CV might not be sufficiently covered by observations from the training set. This makes a LOPO CV very challenging from a pure machine learning perspective.

Summarizing, we consider both a LOIO and LOPO CVs to validate the accuracy of the proposed SOO features. We also target a cheap setting where we can compare how the SOO features are complementary to the existing ELA ones. Hence, for fairness, we consider three feature sets using: (i) solely the SOO features computed with a budget of $100 \times d$ function evaluations, (ii) solely the existing ELA features with a budget of $100 \times d$, and (iii) both the SOO and ELA features each computed with a budget of $50 \times d$ (i.e., $2 \times 50 \times d = 100 \times d$ in total). As previously, a random forest is used.

*3.3.1 High-level properties prediction.* For each cross-validation type, we show in Table 3 the average misclassification error computed over the three dimensions and the different folds. Notice that one model per dimension, per classification task and per feature set is trained and assessed per fold, i.e., $3 \times 8 \times 3 \times 5 = 360$ and $3 \times 8 \times 3 \times 24 = 1728$ models in total for the LOIO CV and the LOPO CV, respectively. First, using a LOIO CV, all features sets show a relatively low misclassification error of about 0.1, i.e., the model is able to correctly classify 90% of the test instances (in average). Using a LOPO CV, it is with no surprise that the misclassification error is higher. However, the SOO features allows to obtain better accuracy than the existing ELA features, except for the prediction of global to local optima contrast (gl.-loc.) characteristic. Interestingly, mixing the SOO with the ELA features does not always provide better results than when using the SOO feature set separately. This can be explained by the high number of features that the random forest model has to accommodate. Such an issue is worth to be investigated in the future using more advanced feature selection techniques, as will be commented later in the conclusion. Remember that the aim of this paper is not to 'beat' the existing ELA features, but to provide evidence that the proposed SOO features can complement them, which is fully in line with our findings.

*3.3.2 Algorithm selection.* In the following, we study the relative accuracy of the SOO features in designing a simple automatic algorithm selection approach. For each dimension and each cross-validation sequence, we build a random forest to predict the relERT of each optimizer, using separately each of the three considered feature sets , i.e., $3 \times 24 \times 10 = 2160$ models for LOPO CV and $3 \times 10 = 450$ models for LOIO CV. For each feature set, the solver with the best predicted relERT is chosen as the output of the algorithm selection task in the testing phase. Hence, we obtain three different automatic *algorithm selectors* (AS), each one corresponding to one feature set. We also consider a baseline approach, called single best solver (SBS), consisting in choosing the portfolio's solver with the best average relERT computed over the training functions.

**Table 3: Mean misclassification error over all dimensions for a budget of $100 \cdot d$. The last row shows the overall average.**

| | LOPO - CV | | | LOIO - CV | | |
|---|---|---|---|---|---|---|
| | ELA | SOO | SOO+ELA | ELA | SOO | SOO+ELA |
| fun.-group | 0.51 | 0.47 | **0.46** | 0.1 | 0.09 | **0.08** |
| multim. | 0.34 | **0.3** | 0.32 | 0.1 | 0.08 | **0.07** |
| gl.-struc. | 0.37 | 0.31 | **0.29** | **0.11** | 0.12 | 0.15 |
| separ. | 0.21 | **0.14** | 0.17 | 0.1 | **0.07** | **0.07** |
| scaling | 0.49 | 0.39 | **0.36** | **0.08** | 0.11 | **0.08** |
| homog. | 0.42 | **0.3** | 0.38 | **0.13** | **0.13** | 0.15 |
| basins | 0.4 | **0.37** | **0.37** | 0.17 | 0.14 | **0.11** |
| gl.-loc. | **0.2** | 0.23 | 0.21 | **0.07** | 0.1 | **0.07** |
| | 0.36 | **0.31** | 0.32 | 0.1 | 0.1 | **0.09** |

In addition to the relERT performance measure, we use ordinal data analysis as a complementary technique to compare the ranks of the obtained algorithm selectors. More precisely, we consider a single-winner election method known as the *Borda count* method [9]. For each experiment (i.e., a testing instance), an approach is given as much points as the number of approaches ranked lower. In our setting, an approach using more function evaluations to hit the target is ranked lower, and unsuccessful runs are ranked the same, regardless of the maximum number of function evaluations performed. Then, the total score of an approach is simply the sum of its points over all experiments. The different approaches can then be compared using their cumulative scores, where the approach with the largest score is considered as the best performing.

Besides having a different objective and theoretical foundation [26], this Borda score has one major difference with the relERT measure, which is the way unsuccessful runs are treated. In fact, by definition, the relERT can introduce a non-negligible bias due to the fact that the maximum number of function evaluations used by an unsuccessful run can highly impact the ERT value. For instance, consider two algorithms that are executed with two different budgets $b$ and $b'$ where $b < b'$. Suppose that the first algorithm (using budget $b$) was successful on solely one instance, whereas the other one was successful on two instances and unsuccessful on the others. Then, it can happen that the first algorithm obtains a much better (lower) ERT value (especially if $b$ is much lower than $b'$). Actually, such a scenario may happen with the BBOB data extracted from COCO, since *not* all algorithms submitted by the different contributors were executed with the same maximum budget. The impact can even be worst when imputing the missing relERT values which is actually one of the challenges all machine learning based ELA approaches are facing with the available COCO data. In our setting, using a Borda score provides an alternative way to appreciate the relative performance of the considered automatic algorithm selectors.

Table. 4 shows the relative performance (relERT and Borda score) of the three feature-based algorithm selectors, together with the SBS, for the challenging LOPO CV setting. Since 5 instances are considered per BBOB function, the maximum (best) Borda score is 15, meaning that an algorithm is better than the three others on all

**Table 4: Borda count and relERT of algorithm selectors using ELA, SOO, SOO+ELA, and the SBSs, for each LOPO fold, i.e., each line corresponds to selecting an algorithm for the actual function instances with the AS trained with the other functions. The winner(s) of the Borda voting method is shown with a box, and the best relERTs, in parenthesis, are shown with a star*.**

| | | $d=2$ | | | | $d=3$ | | | | $d=5$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ELA | SOO | SOO+ELA | SBS | ELA | SOO | SOO+ELA | SBS | ELA | SOO | SOO+ELA | SBS |
| seperable | $f_1$ | **12** (**1***) | 3 (1.33) | 1 (1.7) | 0 (1.97) | 0 (1.95) | 0 (1.95) | 0 (1.95) | 0 (1.95) | 0 (1) | 0 (1) | 0 (1) | 0 (1) |
| | $f_2$ | 0 (1.56) | 0 (1.56) | 0 (1.56) | 0 (1.56) | 0 (2.19) | 0 (2.19) | 0 (2.19) | 0 (2.19) | 0 (2.79) | 0 (2.79) | 0 (2.79) | 0 (2.79) |
| | $f_3$ | 3 (5.71) | **7** (2.59) | **7** (2.59) | 0 (8.27) | 0 (11.26) | 0 (11.26) | 0 (11.26) | 0 (11.26) | 0 (16.52) | **4** (**1.7***) | **4** (**1.7***) | **4** (**1.7***) |
| | $f_4$ | 2 (52.59) | **3** (**5.54***) | 0 (53.17) | 2 (52.59) | 3 (4385.6) | **7** (4383.68) | 2 (4385.6) | 6 (**75.8***) | 0 (2438.43) | 0 (2438.43) | 0 (2438.43) | 0 (2438.43) |
| | $f_5$ | 0 (3.86) | **14** (**1.45***) | 2 (3.36) | 0 (3.86) | 0 (4.55) | 0 (4.55) | 0 (4.55) | 0 (4.55) | 0 (1.56) | 0 (1.56) | 0 (1.56) | 0 (1.56) |
| low or moderate cond. | $f_6$ | 0 (1.9) | **1** (**1***) | **1** (**1***) | **1** (**1***) | 0 (1.07) | 0 (1.07) | 0 (1.07) | 0 (1.07) | **6** (**1.47***) | 3 (2.12) | 3 (2.21) | 0 (2.34) |
| | $f_7$ | **3** (2.87) | **3** (**2.63***) | 0 (3.03) | 0 (3.03) | 0 (4.39) | 0 (4.39) | 0 (4.39) | 0 (4.39) | 0 (1.25) | 0 (1.25) | 0 (1.25) | 0 (1.25) |
| | $f_8$ | 0 (1.57) | 0 (1.57) | 0 (1.57) | 0 (1.57) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (2.36) | **6** (**1.67***) | 3 (2.2) | 0 (2.36) |
| | $f_9$ | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1.12) | 0 (1.12) | 0 (1.12) | 0 (1.12) | 0 (9.52) | 0 (9.52) | 0 (9.52) | 0 (9.52) |
| high cond. and unimodal | $f_{10}$ | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (4.17) | 0 (4.17) | 0 (4.17) | 0 (4.17) |
| | $f_{11}$ | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (7.76) | 0 (7.76) | 0 (7.76) | 0 (7.76) |
| | $f_{12}$ | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1.04) | 0 (1.04) | 0 (1.04) | 0 (1.04) | **3** (**2.04***) | 0 (2.29) | 0 (2.29) | 0 (2.29) |
| | $f_{13}$ | 4 (3.64) | 0 (212.43) | 2 (4.43) | **14** (**1***) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1.8) | **3** (**1.74***) | 0 (1.8) | 0 (1.8) |
| | $f_{14}$ | 1 (3.55) | 1 (3.55) | 3 (3.52) | **8** (**1***) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | **3** (**4.68***) | 0 (5.57) | 0 (5.57) | 0 (5.57) |
| Multi-modal with adequate global struct. | $f_{15}$ | 3 (1.5) | 3 (1.5) | **5** (**1.24***) | **5** (2.39) | 0 (10.98) | 0 (10.98) | 0 (10.98) | 0 (10.98) | 0 (1) | 0 (1) | 0 (1) | 0 (1) |
| | $f_{16}$ | 1 (11.3) | **3** (**10.52***) | 1 (11.3) | 1 (11.3) | 0 (17.91) | 0 (17.91) | 0 (17.91) | 0 (17.91) | 0 (2.31) | 0 (2.31) | 0 (2.31) | 0 (2.31) |
| | $f_{17}$ | 0 (7.92) | 5 (**4.1***) | **8** (5.93) | 0 (7.92) | 0 (11.32) | 0 (11.32) | 0 (11.32) | 0 (11.32) | 0 (1.5) | 0 (1.5) | 0 (1.5) | 0 (1.5) |
| | $f_{18}$ | 0 (18.18) | **9** (**12.98***) | 0 (18.18) | 0 (18.18) | 0 (7.75) | 0 (7.75) | 0 (7.75) | 0 (7.75) | 0 (1) | 0 (1) | 0 (1) | 0 (1) |
| | $f_{19}$ | 0 (18.87) | **1** (**1***) | **1** (**1***) | **1** (**1***) | 0 (15.67) | 0 (15.67) | 0 (15.67) | 0 (15.67) | 0 (3.31) | 0 (3.31) | 0 (3.31) | 0 (3.31) |
| Multi-modal with weak global struct. | $f_{20}$ | 0 (2.75) | 0 (2.75) | 0 (2.75) | 0 (2.75) | 0 (6.33) | 0 (6.33) | 0 (6.33) | 0 (6.33) | **5** (2.17) | **5** (**1.67***) | 2 (2.92) | 3 (1.99) |
| | $f_{21}$ | 0 (1.38) | 0 (1.38) | 0 (1.38) | 0 (1.38) | 0 (1.32) | 0 (1.32) | 0 (1.32) | 0 (1.32) | 0 (23.43) | 0 (23.43) | 0 (23.43) | 0 (23.43) |
| | $f_{22}$ | 0 (1.33) | 0 (1.33) | 0 (1.33) | 0 (1.33) | 0 (1.76) | 0 (1.76) | 0 (1.76) | 0 (1.76) | 0 (4.99) | 0 (4.99) | 0 (4.99) | 0 (4.99) |
| | $f_{23}$ | 0 (1.59) | 0 (1.59) | 0 (1.59) | 0 (1.59) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) | 0 (1) |
| | $f_{24}$ | 2 (**21.42***) | **3** (108.34) | 1 (**21.42***) | 0 (**21.42***) | 0 (2.55) | 0 (2.55) | 0 (2.55) | 0 (2.55) | 0 (1) | 0 (1) | 0 (1) | 0 (1) |
| | $\sum$ (**avg**) | 31 (7.02) | **56** (15.96) | 32 (**6.09***) | 32 (6.21) | 3 (187.28) | **7** (187.2) | 2 (187.28) | 6 (7.71*) | 17 (105.71) | **21** (**105.12***) | 12 (105.2) | 7 (105.17) |

testing instances. For a 0 Borda count, the corresponding approach is never able to outperform any other approach on any of the testing instances (regardless of whether it is successful or not to hit the target). Similarly, when the relERT is 1, the considered approach corresponds to the best performing on the BBOB function.

The SOO-based AS shows equal or better Borda scores than the ELA-based AS on all functions with four exceptions ($f_1$ at $d = 2$, and $f_6$, $f_{12}$, $f_{14}$ at $d = 5$). The relERT is in accordance with the Borda score with some notable exceptions that are either in favor of the SOO-based AS ($f_{24}$ at $d = 2$) or of the ELA-based AS ($f_7$ at $d = 2$ and $f_{20}$ at $d = 5$). We can also notice that the relERT obtained by the SOO-based AS for $f_{13}$ (Sharpe Ridge) at $d = 2$ and $f_{24}$ (Lunacek bi-Rastrigin) are extremely high, while the Borda score is relatively good. This has the effect of decreasing the average relERT (over all function folds). A fine-grained look at the results revealed that this is due to the some bias in the ERT computation when counting the function evaluations for unsuccessful runs.

Over all BBOB functions, the SOO-based AS obtains a better Borda score at every dimension. The most notable improvements are observed at dimension $d = 2$ and to a less extent at dimension $d = 5$, whereas all approaches have similar performance for $d = 3$. We can also see that mixing the SOO features with the ELA features does not necessarily lead to improvements, which we similarly attribute to the high number of features and the need of using more advanced feature selection techniques. Finally, we can remark that whenever the relERT of an AS is 1, the relERT of the SBS is also 1. Ideally, one would like an AS to have a relERT of 1 for every function, which is of course extremely hard to achieve in a LOPO scenario. Moreover, we found that the SBS does not change much over the different folds, with **HCMA** and **HMLSL** being extremely effective on most instances. Overall, we are able to observe a slight improvement over the SBS, even-though for few instances ($f_{13}$ at $d = 2$ and $f_4$ at $d = 3$), a bad prediction on one single instance makes a significant decrease in the relERT.

To further illustrate the benefits of the SOO features, Fig. 7 show the proportion of instances that an algorithm is able to solve at the target precision value, as a function of the number of function evaluations used to run the algorithm, which is inspired by the empirical cumulative distribution functions (ECDF). However, we also include the behavior of the (artificial) algorithm implied by the SOO-based ASs (remember that each LOPO CV fold per function implies a different AS). Rather than a single numerical value like the relERT or the Borda score, this allows us to provide a more complete idea on the profile of different approaches, i.e., the more a curve is on the left, the better is the algorithm. For better clarity, we only show the algorithms from the portfolio allowing to obtain the most interesting and representative profiles, i.e., none of the

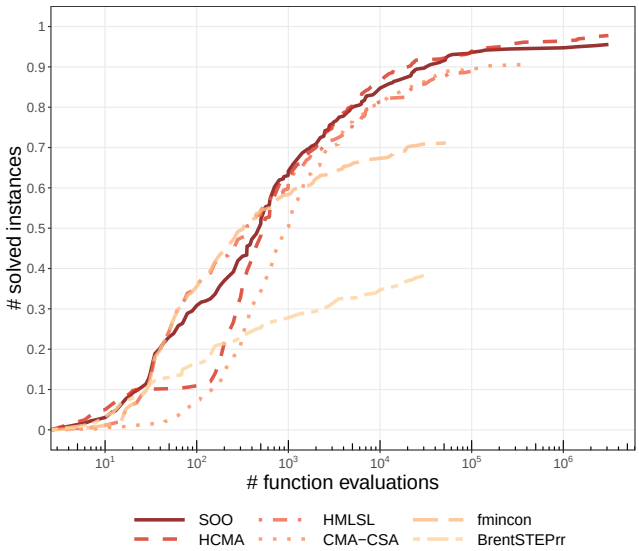


**Figure 7: Proportion, over all the $24\times5$ functions/LOPO-folds and instances, of solved instances (w.r.t the $10^{-2}$ target value) by different algorithms as a function of the number of function evaluations (per instance) including the SOO-based ASs.**

removed ones can dominate any of the depicted algorithms at any point. Two main observations can be formulated from Fig. 7.

First, there is no algorithm that is better than all the others for any number of function evaluations. However, **HCMA** appears to be extremely efficient when given more function evaluations, although it cannot reach a proportion of 1, i.e., some instances are very difficult to solve, even when provided a high budget. However, for some instances, other algorithms like **HMLSL** and **fmincon**, need only a restricted number of function evaluations. Second, the SOO-based AS is able to approach the best of these three algorithms, which is typically the expectation behind an ELA approach. Nonetheless, there is still room for a small improvement, which can likely be achieved in the future by improving the relERT predictions. Since the portfolio contains relatively efficient algorithms, we expect the task to be anyway very challenging in a LOPO setting, given the diversity of the BBOB feature space.

## 4 CONCLUSION AND PERSPECTIVES

In this paper, we introduced a novel set of features based on the SOO global optimizer. Guided by its exploitation-exploration design, SOO implies a specific sampling of the variable space using a tree structure. Using a simple visualization tool, the SOO tree is found to be informative about the landscape. The shape of the tree, as well as the fitness of nodes all along the different levels of the tree, are then summarized using a number of relatively simple statistics constituting the proposed features. Based on an exploratory landscape analysis methodology, the relevancy of the proposed features is demonstrated based on a systematic analysis of their relative importance and accuracy when conducting a number of challenging tasks. In particular, it is shown that the SOO features perform relatively well for high-level property prediction and algorithm selection. We thus conclude that the proposed SOO features are good candidates for more advanced ELA developments. In fact, we left open a number of open questions that we summarize in the following, while providing some general perspectives ranging from landscape visualization, to feature definition and algorithm design.

*Landscape tree visualization.* We believe that visualization helps providing new insights in our fundamental understanding of the landscape characteristics. As such, more advanced visualization tools are worth to be investigated in the future in order to better render the structure of the SOO tree. We can for instance notice that the 2D layout used for visualizing the SOO tree implies a sort of deformation with respect to the initial positions of nodes in the variable space. One idea would be to take into account the position (in the variable space) of nodes in the tree layout specification, or to use other more specific layouts from the specialized literature. For two and three dimensions, a direct mapping of cells centers, in the variable space, to the coordinates of tree nodes in a 2D or 3D layer could be feasible in a relatively simple manner. For higher dimensions, the relative distance between nodes in the variable space could help implying a better rendering when computing the nodes layout coordinates. The fitness value of nodes is also one information that should be better rendered. One idea could be to define a sort of basins of attractions within the tree and to enable a better visual identification. Beside allowing a better visual rendering, one original idea would be to explore the possibility of computing some properties of the produced image (e.g., density, space filling, intensity) in order to enable the computation of some landscape features, or even conducting some pattern recognition tasks directly on the produced image.

*Tree dynamics features.* Among the five groups of proposed features, one group is dedicated to summarize the dynamics of the tree across SOO iterations. The designed features are relatively simple as they focus on the evolution of the number of splits and the successive heights in the evolving tree. One future idea would be to rely on a more advanced tracking of the whole shape of the tree. For instance, this can be enabled by leveraging the statistics from the first two groups, about tree shape and fitness, and summarizing their distribution over the successive algorithm iterations.

*Other feature groups.* Based on the foundations of the SOO algorithm design, it should be clear that a number of other meaningful features can be thought. Firstly, mertics and distances about trees from the specialized literature can be considered. However, the implied computational complexity can be an issue since this is in itself a hot issue when dealing with tree spaces. It is in fact to notice that all our features have either linear or quadratic (in the tree size) complexity, and so are the SOO algorithmic components. This is to contrast with some existing ELA features, e.g., those related to cell mapping, that are extremely expensive and memory consuming to compute as the dimension is scaled, even for the default setting of their parameters, e.g., the number of required cells. Secondly, other metrics related to the distribution of sampled points in the variable space can be considered, which would eventually lead to other feature groups with a different nature that those described in this paper. One idea is for instance to use clustering techniques to

grasp the structure of points sampled by SOO. Another idea would be to summarize the exploration-exploitation trade-off that the SOO algorithm is implying, in order to better grasp the difficulty of the landscape. This can be achieved by an empirical or theoretical (from the proofs in [27]) analysis rendering the expected behavior of the SOO tree expansion on some exemplified functions considered as a reference.

*SOO sample as input in flacco.* The existing ELA features, as provided in the flacco library, need an initial sample of points from the landscape. In all existing ELA studies, this is achieved by Latin Hypercube Sampling techniques, which explains why we used separate budgets when computing the SOO and the existing ELA features. Although we took care of considering a reduced budget setting in our comparative analysis, the impact of budget is left open for future investigations. More importantly, an alternative design idea is to use the SOO sample as an input to the existing ELA ones. In fact, besides eventually reducing the overall budget which might be necessary in an expensive setting, the accuracy of some existing ELA features can highly benefits from such a design, e.g., features based on meta-modelling.

*Feature selection.* As mentioned in our analysis, applying feature selection techniques should very likely lead to improved algorithm selectors as observed in previous ELA approaches. On the other hand, feature selection can be used as a tool to discriminate between features for some specific target tasks. However, as the size of the ELA feature set increases, standard (greedy) techniques could be inaccurate and/or prohibitive. More generally, feature selection is to be handled with special care, and advanced efficient strategies to identify the most interesting and efficient subset(s) of features are still to be thought in the future, specifically with respect to the target ELA tasks.

*Per instance algorithm selection / other benchmarks and algorithms.* As discussed in our ELA study, and also as highlighted in a number of existing ones, we think that model validation using the BBOB functions and the corresponding data from COCO, is very challenging, and sometimes problematic given the initial purpose of the BBOB tesbed and the non-uniformity of the COCO data. For instance, in a real-world setting, where it often holds that different instances of the same problem have to be solved, other validation methodologies could be considered, which could for example be related to the so-called per-instance algorithm configuration problem. We also need more data about algorithm executions. Some algorithms from COCO have only one execution per instance. We believe that other more focused studies considering a limited type of problems, more instances, and more comprehensive data about algorithm performance, could help to better highlight the open challenges when adopting an ELA approach.

## REFERENCES


[1] Asma Atamna. 2015. Benchmarking IPOP-CMA-ES-TPA and IPOP-CMA-ES-MSR on the BBOB Noiseless Testbed. In *GECCO Companion*. 1135–1142.

[2] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. 2013. Benchmarking the local metamodel CMA-ES on the noiseless BBOB'2013 test bed. In *GECCO Companion*. 1225–1232.

[3] Anne Auger and Nikolaus Hansen. 2005. A restart CMA evolution strategy with increasing population size. In *CEC*. 1769–1776.

[4] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuss. 2012. Algorithm Selection Based on Exploratory Landscape Analysis and Cost-sensitive Learning. In *GECCO*. ACM, Philadelphia, USA, 313–320.

[5] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Claus Weihs. 2012. Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation. *Evolutionary Computation* 20, 2 (2012), 249–275.

[6] Michael G.B. Blum and Olivier François. 2005. On statistical tests of phylogenetic tree imbalance: The Sackin and other indices revisited. *Mathematical Biosciences* 195, 2 (2005), 141 – 153.

[7] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[8] D.H. Colless. 1982. [Review of] Phylogenetics: the theory and practice of phylogenetic systematics. *Syst. Zool.* 31 (1982), 100–104.

[9] J. C. de Borda. 1781. Memoire sur les élections au scrutin. *Historie de l'Academie Royale des Sciences, Paris, France* (1781).

[10] Bilel Derbel and Philippe Preux. 2015. Simultaneous optimistic optimization on the noiseless BBOB testbed. In *CEC*. 2010–2017.

[11] Youssef Hamadi, Eric Monfroy, and Frédéric Saubion (Eds.). 2012. *Autonomous Search*. Springer.

[12] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup*. Research Report RR-6828. INRIA. https://hal.inria.fr/inria-00362649

[13] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions*. Research Report RR-6829. INRIA. https://hal.inria.fr/inria-00362633

[14] F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *Learning and Intelligent OptimizatioN*. 507—523.

[15] Waltraud Huyer and Arnold Neumaier. 2009. Benchmarking of MCS on the Noiseless Function Testbed. In *GECCO Companion*.

[16] D.R. Jones, C.D. Perttunen, and B.E. Stuckman. 1993. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications* 79, 1 (1993), 157–181.

[17] Pascal Kerschke. 2017. Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package flacco. *arXiv e-prints* (Aug 2017), arXiv:1708.05258.

[18] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. 2019. Automated Algorithm Selection: Survey and Perspectives. *Evolutionary Computation* 27, 1 (2019), 3–45.

[19] Pascal Kerschke and Heike Trautmann. 2019. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation* 27, 1 (2019), 99–127.

[20] Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, Hernán E. Aguirre, and Kiyoshi Tanaka. 2017. Towards Landscape-Aware Automatic Algorithm Configuration: Preliminary Experiments on Neutral and Rugged Landscapes. In *EvoCOP*. 215–232.

[21] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. 2013. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems 26*. 431–439.

[22] KatherineM. Malan and AndriesP. Engelbrecht. 2014. Fitness Landscape Analysis for Metaheuristic Performance Prediction. In *Recent Advances in the Theory and Application of Fitness Landscapes*, Hendrik Richter and Andries Engelbrecht (Eds.). Emergence, Complexity and Computation, Vol. 6. Springer, 103–132.

[23] Andy McKenzie and Mike Steel. 2000. Distributions of cherries for two models of trees. *Mathematical Biosciences* 164, 1 (2000), 81 – 92.

[24] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *GECCO*. 829–836.

[25] Olaf Mersmann, Mike Preuss, and Heike Trautmann. 2010. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In *PPSN XI*. 73–82.

[26] Olaf Mersmann, Mike Preuss, Heike Trautmann, Bernd Bischl, and Claus Weihs. 2015. Analyzing the BBOB Results by Means of Benchmarking Concepts. *Evolutionary Computation* 23, 1 (2015), 161–185.

[27] Rémi Munos. 2011. Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness. In *Advances in Neural Information Processing Systems*. 783–791.

[28] Mario A. Muñoz, Yuan Sun, Michael Kirley, and Saman K. Halgamuge. 2015. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences* 317, 0 (2015), 224 – 245.

[29] O. Mersmann; T. Tušar; N. Hansen; A. Auger and D. Brockhoff. 2016. COCO: A platform for Comparing Continuous Optimizers in a Black-Box Setting. *ArXiv e-prints* (2016).

[30] László Pál. 2013. Comparison of multistart global optimization algorithms on the BBOB noiseless testbed. In *GECCO Companion*. 1153–1160.

[31] Petr Posík and Petr Baudis. 2015. Dimension Selection in Axis-Parallel Brent-STEP Method for Black-Box Optimization of Separable Continuous Functions. In *GECCO*. 1151–1158.

[32] Philippe Preux, Rémi Munos, and Michal Valko. 2014. Bandits attack function optimization. In *CEC*. 2245–2252.

[33] John R. Rice. 1976. The Algorithm Selection Problem. Advances in Computers, Vol. 15. Elsevier, 65 – 118.
[34] Hendrik Richter and Andries Engelbrecht (Eds.). 2014. *Recent Advances in the Theory and Application of Fitness Landscapes*. Springer.
[35] M.J. Sackin. 1972. "Good" and "Bad" phenograms. *Syst. Zool.* 21, 2 (1972), 225.
[36] Kate Amanda Smith-Miles. 2008. Towards insightful algorithm selection for optimisation using meta-learning concepts. In *International Joint Conference on Neural Networks*. IEEE, Hong Kong, 4118–4124.
[37] Kate A. Smith-Miles. 2009. Cross-disciplinary Perspectives on Meta-learning for Algorithm Selection. *ACM Comput. Surv.* 41, 1, Article 6 (Jan. 2009), 25 pages.
[38] Zsolt Ugray, Leon S. Lasdon, John C. Plummer, Fred Glover, James P. Kelly, and Rafael Martí. 2007. Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization. *INFORMS Journal on Computing* 19, 3 (2007), 328–340.
[39] Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. 2010. Hydra: Automatically Configuring Algorithms for Portfolio-based Selection. In *Conference on Artificial Intelligence*. 210–216.