Information Sciences 317 (2015) 224–245

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges

Mario A. Muñoz [a,b,*], Yuan Sun [b], Michael Kirley [c], Saman K. Halgamuge [b]

[a] School of Mathematical Sciences, Monash University, Clayton, Victoria 3800, Australia
[b] Department of Mechanical Engineering, The University of Melbourne, Parkville, Victoria 3010, Australia
[c] Department of Computer and Information Systems, The University of Melbourne, Parkville, Victoria 3010, Australia

ARTICLE INFO

Article history:
Received 18 November 2014
Received in revised form 17 March 2015
Accepted 1 May 2015
Available online 7 May 2015

Keywords:
Algorithm selection
Black-box continuous optimization
Empirical performance models
Exploratory landscape analysis
Performance prediction
Problem hardness measures

ABSTRACT

Selecting the most appropriate algorithm to use when attempting to solve a black-box continuous optimization problem is a challenging task. Such problems typically lack algebraic expressions, it is not possible to calculate derivative information, and the problem may exhibit uncertainty or noise. In many cases, the input and output variables are analyzed without considering the internal details of the problem. Algorithm selection requires expert knowledge of search algorithm efficacy and skills in algorithm engineering and statistics. Even with the necessary knowledge and skills, success is not guaranteed.

In this paper, we present a survey of methods for algorithm selection in the black-box continuous optimization domain. We start the review by presenting Rice's (1976) selection framework. We describe each of the four component spaces – problem, algorithm, performance and characteristic – in terms of requirements for black-box continuous optimization problems. This is followed by an examination of exploratory landscape analysis methods that can be used to effectively extract the problem characteristics. Subsequently, we propose a classification of the landscape analysis methods based on their order, neighborhood structure and computational complexity. We then discuss applications of the algorithm selection framework and the relationship between it and algorithm portfolios, hybrid meta-heuristics, and hyper-heuristics. The paper concludes with the identification of key challenges and proposes future research directions.

1. Introduction

The objective of an optimization problem is to improve a measure of performance or cost – the output variable – by adjusting the values of the input variables. Typically, the optimization problem is represented as a function that maps the inputs to the output, subject to constraints. When both the input and output variables are real numbers, the problem is referred to as a continuous optimization problem. Such problems are common in science, engineering, finance, and other fields [81].

Many continuous optimization problems lack algebraic expressions and may not even have a precise goal. Topologically, the problem may present local and global optima or discontinuities where it is not possible to calculate derivative information. The problems frequently incorporate dependencies between input variables, and large output fluctuations between

* Corresponding author at: School of Mathematical Sciences, Monash University, Clayton, Victoria 3800, Australia.
E-mail address: mario.munoz@monash.edu (M.A. Muñoz).

http://dx.doi.org/10.1016/j.ins.2015.05.010
0020-0255/

adjacent input values – characteristics that are often unknown *a priori*. Furthermore, the problem may involve simulations or experiments, which may be computational expensive or resource intensive. The input and output variables are often analyzed without considering the internal details of the problem. In other words, the problem is modeled as a black-box.

A search algorithm is typically used to solve a black-box continuous optimization problem (BCOP). Given the fact that the number of algorithms introduced into the optimization community has increased over the past decades, it is extremely difficult for practitioners to be familiar with the specifics of all the algorithms [59]. Compounding the issue, is the diverse array of models used by algorithms to generate candidate solutions. In this context, the model is a set of assumptions describing the relationship between input and output variables. Nevertheless, the problem characteristics may break these assumptions, degrading the algorithm performance [50,159]. Unfortunately, we have a limited theoretical understanding of the strengths and weaknesses of most algorithms [93,96,119,111,145]. Consequently, selecting the most efficient algorithm for a given BCOP is non-trivial and is at best cumbersome [147]. Algorithm selection requires expert knowledge of search algorithms, and skills in algorithm engineering and statistics [15]. Even then, failure is still an option.

A number of approaches, such as algorithm portfolios, hybrid meta-heuristics, hyper-heuristics, parameter tuning and control methods, have been proposed to circumvent the algorithm selection challenge [4,8,13,18,35,41,63,64,115]. However, these approaches disregard the similarities between the current problem and previously observed problems [23]. An alternative is to construct a map between the problem and algorithm spaces [125]. Due to the complexity of these two spaces, the map can be constructed between measures of the problem characteristics and an algorithm performance measure [125], using a machine learning model [137]. This approach has been successful in a number of problem domains [2,9,39,44–46,61,62,65,69,70,73–77,87–89,136–139,141,148,157,166–169] including BCOPs [1,14,102], where Exploratory Landscape Analysis (ELA) [91] methods are used to measure the problem characteristics.

In this paper, we present a survey of methods for algorithm selection for BCOPs. The paper is organized as follows: In Section 2, we describe the algorithm selection framework proposed by Rice in [125], which lays down the conceptual foundations for the problem–algorithm map. Each of the four component spaces – problem, algorithm, performance and characteristic – are described in terms of requirements for black-box continuous optimization problems in subsequent sections. In Section 3, we formally define a BCOP and problem characteristics, employing the fitness landscape concept. In Section 4, we briefly discuss different classes of search algorithms. As such, this section illustrates the diversity of algorithms available to practitioners. In Section 5, we introduce the methods used to measure algorithm performance. In Section 6, we present a classification and a summary of well known ELA methods for the analysis of continuous fitness landscapes. We then present implementations of the algorithm selection framework for BCOPs, and describe related approaches in Section 7. Finally, in Section 8 we propose future research directions.

2. The algorithm selection framework

The algorithm selection problem (ASP) is defined as follows[1]: Let $\mathcal{F}$ be a problem space or domain, such as continuous optimization. Let $\mathcal{A}$ be the algorithm space, which is a set of algorithms that can be applied to the problems in $\mathcal{F}$. For a given problem $f \in \mathcal{F}$, the objective is to find an algorithm $\alpha_o \in \mathcal{A}$ that minimizes $\rho(f, \alpha)$, which is a measure of the cost of using an algorithm $\alpha \in \mathcal{A}$ to solve f. Perhaps Rice was the first to describe the ASP in his 1976 paper [125]. He introduced the framework illustrated in Fig. 1 to solve the ASP [137].

The framework has four components. The first one, $\mathcal{F}$, cannot be properly defined without using complex mathematical notation [126]. Although $\mathcal{F}$ has infinite cardinality, it excludes unsolvable problems such as a random function. Furthermore, $\mathcal{F}$ is high dimensional due to the large number of problem characteristics [125]. The second component of the framework is $\mathcal{A}$, which potentially has an infinite cardinality. However, it is impractical to consider all reported algorithms [125]. Instead, $\mathcal{A}$ contains the smallest set of algorithms, such that each one of them solves a large subset of problems in $\mathcal{F}$ with the best possible performance [6]. These algorithms should be complementary – they solve different types of problems – and robust – their accuracy and precision is scientifically demonstrable. The third component of the framework is the performance space, $\mathcal{P} \subset \mathbb{R}$. It is the set of feasible values of $\rho(f, \alpha)$. The performance measure represents the accuracy, speed, or other algorithmic requirements.

The ASP is ill-defined due to the complexity and size of $\mathcal{F}$ and $\mathcal{A}$. Therefore, formal solutions to the ASP may not exist. To simplify the problem, Rice introduced the features or characteristics space, $\mathcal{C} \subset \mathbb{R}^m$, as the fourth component of his framework. Characteristics are domain dependent, and provide order to $\mathcal{F}$ by imposing a low dimensional space [126,154]. Furthermore, characteristics must be measurable, expose the complexities of f, and provide knowledge about the strengths and weaknesses of α [137]. Hence, selecting the characteristics is a key step in the application of the framework [125].

The framework is conceptually rich, although it lacks a clear implementation path for the selection mapping $S : \mathcal{C} \rightarrow \mathcal{A}$. Consequently, the framework lacked extensive application until machine learning models – in this context known as meta-models – became a suitable implementation approach [137]. We will further discuss the meta-models and their implementation in Section 7.

[1] We have modified Rice's notation to use f as function. In the original paper, x is a problem, $f(x)$ is a feature, A is an algorithm, $p(A, x)$ is a performance measure, $\mathcal{P}$ is the problem space, and $\mathcal{F}$ is the feature/characteristics space.

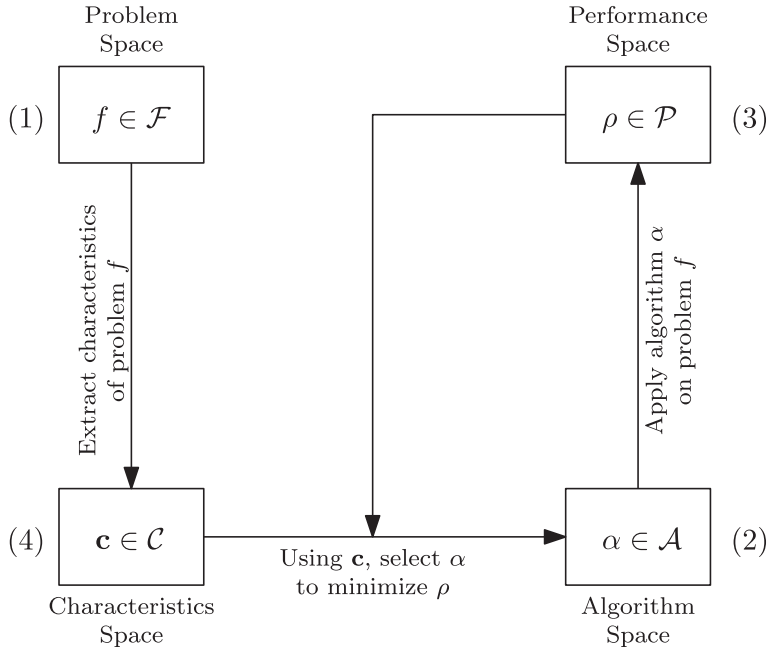


Fig. 1. Diagram of the algorithm selection framework proposed by Rice in his 1976 paper [125,137]. The framework has four components: the problem, algorithm, performance and characteristic spaces.

3. Problem space: continuous optimization and fitness landscapes

The first component of the algorithm selection framework is the problem space, $\mathcal{F}$, defined in Section 2 as the set of all the possible problems in the domain. The goal in a continuous optimization problem is to minimize or maximize the function $f : \mathcal{X} \mapsto \mathcal{Y}$ where $\mathcal{X} \subset \mathbb{R}^D$ is a bounded and compact space known as the input space, $\mathcal{Y} \subset \mathbb{R}$ is known as the output space, and $D \in \mathbb{N}^+$ is the dimensionality of the function. From now on we shall use problem and function interchangeably and assume minimization without loss of generality.

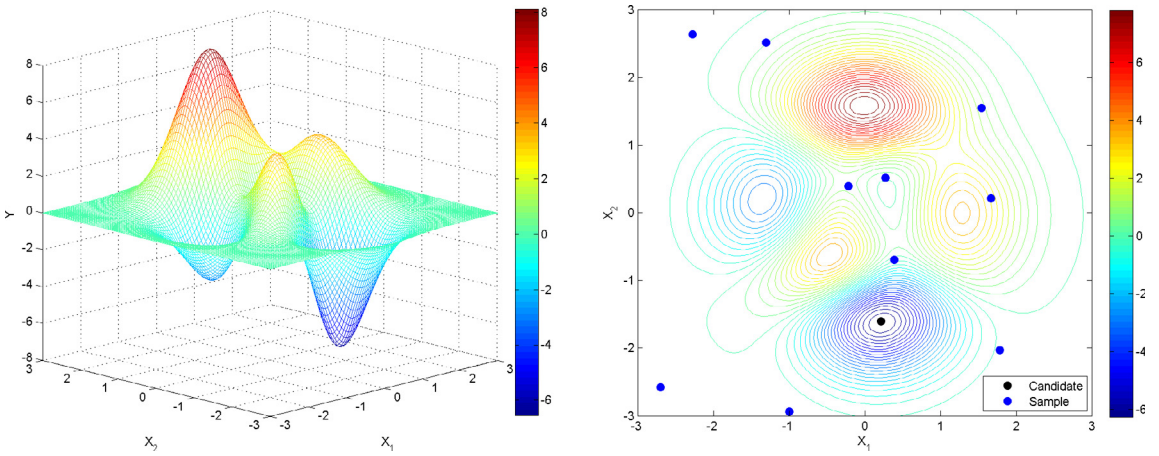
Let $\mathbf{x} \in \mathcal{X}$ be a candidate solution with a cost or fitness of $y \in \mathcal{Y}$ such that $y = f(\mathbf{x})$. We call the subset of candidates the input sample, $\mathbf{X}$, and their cost values the output sample, $\mathbf{Y}$. A BCOP is a problem such that it cannot be described with simple mathematical expressions, lacks of useful or computable first and second derivatives, or has information that is not explicit in the problem description, such as randomness or uncertainty. For these problems, the inputs and the outputs of the function are analyzed without considering the function's internal workings. Therefore, a solution can only be found by sampling the input space.

An optimal solution, $\mathbf{x}_o \in \mathcal{X}, y_o = f(\mathbf{x}_o)$, is defined as the candidate for which all the elements in $\mathcal{X}$ have higher cost, i.e., $\{\mathbf{x}_o : \forall \mathbf{x} \in \mathcal{X}, y \geqslant y_o\}$. In practice, it is neither fundamental nor feasible to find $\mathbf{x}_o$. Hence, we define a target solution, $\mathbf{x}_t, y_t = f(\mathbf{x}_t)$, as the candidate for which the difference in cost with $\mathbf{x}_o$ is less or equal than an acceptable value, e_t, i.e., $\{\mathbf{x}_t : \mathbf{x}_t \in \mathcal{X}, y_t - y_o \leqslant e_t\}$. Alternatively, $\mathbf{x}_t$ can be defined as the candidate that offers the maximum improvement over a previously known cost if y_o is unknown.

The only restriction on the sampling procedure used to find $\mathbf{x}_t$ is that it generates candidates inside of $\mathbf{X}$. Although a random or a grid search are valid sampling procedures, neither of them are likely to find $\mathbf{x}_t$ in finite time. A better way to look for $\mathbf{x}_t$ is to use a search algorithm, which is a systematic procedure that takes previous candidates and their costs as inputs, and generates new and hopefully better candidates as outputs. For this purpose, the search algorithm maintains a simplified model of the relationship between candidates and costs. The model assumes that the problem has an exploitable structure; hence, the algorithm can infer details about the structure of the problem by collecting information during the run. This structure is known as the fitness landscape [122].

To understand the concept behind fitness landscapes, consider a function with $D = 2$. The fitness landscape can be thought of as a surface in a three dimensional space[2] composed of ridges, valleys, plateaus and basins as illustrated on Fig. 2. The local and global optima are located at the lowest points of this surface [153]. In this context, the goal of the search algorithm is to find a target solution, storing information about the structure of the surface as it progresses to update its model. The way in which the model infers the fitness landscape has a direct impact on its performance.

[2] Two dimensions correspond to $\mathcal{X}$ and one more corresponds to $\mathcal{Y}$.



(a) Surface view (b) Contour view

Fig. 2. Fitness landscape of a function with $D = 2$. (a) Illustrates the landscape as a three dimensional surface composed of ridges, valleys, plateaus and basins. The local and global optima are located at the lowest points of this surface. (b) Shows a contour view of the landscape and the location of a candidate solution, $\mathbf{x} \in \mathcal{X}$ and an input sample, $\mathbf{X} \subset \mathcal{X}$. The color bars indicate the value of the cost. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Formally, a fitness landscape $\mathcal{L}$ for a function f is the tuple $\mathcal{L} = (\mathcal{X}, f, d)$, where d is a distance metric that quantifies the similarity between candidates. In continuous optimization, d is often the Euclidean distance. Similar candidates are those whose distance between them is less than a threshold [118]. These candidates form a neighborhood, $\mathcal{N}_r(\mathbf{x}) \subset \mathcal{X}$, defined as follows:

$$\mathbf{x}_i \in \mathcal{N}_r(\mathbf{x}) \iff d(\mathbf{x}, \mathbf{x}_i) \leqslant r \tag{1}$$

where r is the radius of a hypersphere centered on $\mathbf{x}_i$. Empirical studies suggest that the volume of $\mathcal{N}_r$ and the search algorithm performance are correlated [36]. Furthermore, as D increases so does the size of $\mathcal{X}$ and the volume of $\mathcal{N}_r$, resulting in loss of algorithm performance [80], an effect known as the curse of dimensionality [60].

Using the definitions of fitness landscape and neighborhood presented above, we provide formal definitions for local and global optima. A local optimum in a landscape $\mathcal{L}$ is a candidate $\mathbf{x}_l \in \mathcal{X}$ such that $y > y_l$ for all $\mathbf{x} \in \mathcal{N}_r(\mathbf{x}_l)$. The set of local optima can be denoted as $\mathcal{X}_l \subset \mathcal{X}$. A global optimum in a landscape $\mathcal{L}$ is a candidate $\mathbf{x}_o \in \mathcal{X}_l$ such that $y_l \leqslant y_o$ for all $\mathbf{x}_l \in \mathcal{X}_l$. The set of global optima can be denoted as $\mathcal{X}_o \subseteq \mathcal{X}_l$. These definitions will come in handy when we describe the different characteristics of a function.

3.1. Modality and smoothness

Some characteristics of the fitness landscape can be described using the cardinality of both $\mathcal{X}_l$ and $\mathcal{X}_o$. An unimodal landscape (Fig. 3a) is defined as the landscape with $|\mathcal{X}_o| = |\mathcal{X}_l| = 1$. A multimodal landscape (Fig. 3b) is defined as the landscape with $|\mathcal{X}_l| > 1$. Although a multimodal landscape is thought to be more difficult to search than an unimodal landscape, this intuition is sometimes unfounded [92,162].

A closely related concept to multimodality is smoothness, which refers to the magnitude of change in fitness within a neighborhood. A landscape can be informally classified as rugged, smooth or neutral. Rugged landscapes (Fig. 3c) have large fluctuations between neighbors, presenting several local optima and steep ascents and descents. Neutral landscapes (Fig. 3d) have large flat areas or plateaus, where changes in the input fail to generate significant changes in the output. Due to the characteristics of both rugged and neutral landscapes, both gradient and correlation data provide scarce information. Thus, it is difficult to assert whether a particular area of the landscape is worth exploring [162].

The pattern of the set of the local optima, $\mathcal{X}_l$, constitutes the global structure of the landscape, which can be used to guide the search for $\mathbf{x}_t$. If this pattern is smooth (Fig. 3e), it provides potentially exploitable information for an algorithm. If this pattern is rugged or inexistent, finding an optimum can be challenging, perhaps impossible [92]. It is also possible to find problems that exhibit deceptiveness, i.e., the global structure and gradient information lead the algorithm away from the optima (Fig. 3f), rendering the optimization algorithm less efficient than a random search or exhaustive enumeration [162]. In addition, it is possible that the fitness landscape possesses opposite properties at different locations, e.g., highly rugged in some regions, while neutral in others. These landscapes are said to be globally heterogeneous [92].

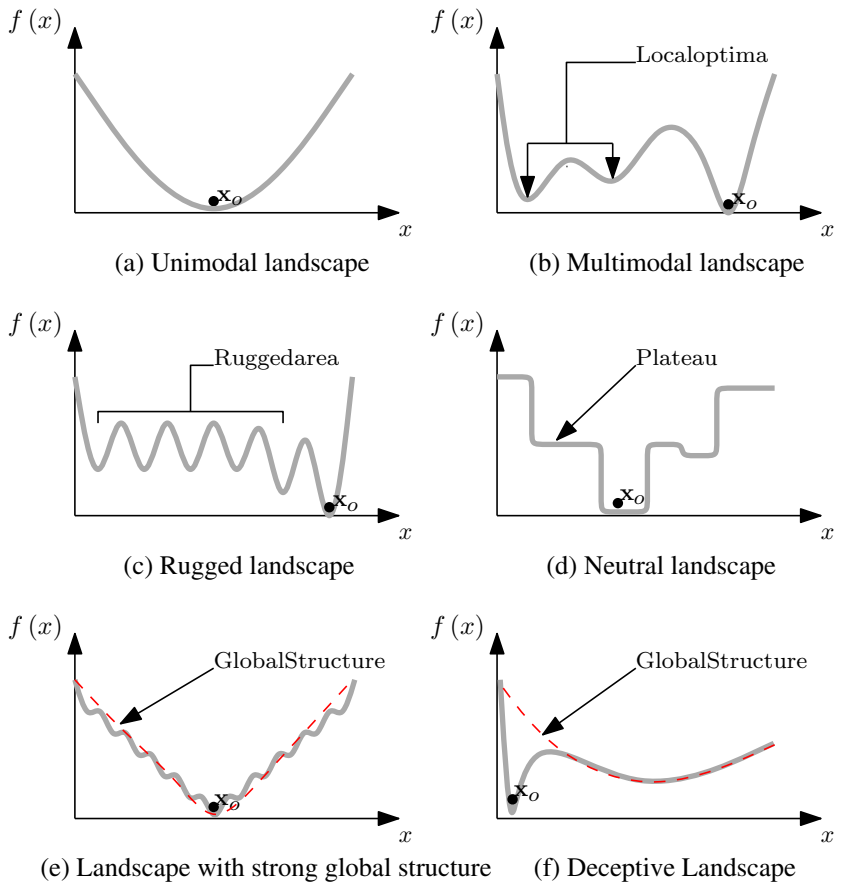


Fig. 3. Plots of six typical function in one dimension, where each plot depicts a specific landscape characteristic.

3.2. Basins of attraction

In dynamical systems, any system that evolves towards a steady state is said to be drawn to an attractor [171]. This idea can be used in the context of continuous optimization, if we consider a local optimum to be a steady state. For this purpose, define a local search as a function $\mu : \mathcal{X} \to \mathcal{X}_l$, and the basin of attraction of a local optimum, $\mathbf{x}_l$, as the subset of $\mathcal{X}$ in which the local search converges towards $\mathbf{x}_l$, i.e., $\mathcal{B}(\mathbf{x}_l) = \{\mathbf{x} : \mu(\mathbf{x}) = \mathbf{x}_l\}$, where $\mathbf{x}$ is the initial candidate and $\mu(\mathbf{x})$ is the local optimum found.

The shape and distribution of the basins of attraction are key characteristics of the fitness landscape. Ideally, a problem has isotropic or spherical basins (Fig. 4a). However, real-world problems may present anisotropic or non-spherical basins (Fig. 4b). Usually, anisotropic basins are the result of non-uniform variable scaling. This means that the change in cost resulting from modifying one of the variables is marginal compared to the change resulting from modifying any of the other variables. A successful search in such landscapes requires small steps in one variable and large steps in others. Anisotropic basins might be non-elliptical. Therefore, the problem cannot be broken down into easier problems of lower dimensionality [92]. Such non-separable problems may be significantly difficult for certain algorithms [50], and increase the computational cost [80].

Both the shape and the size of basins of attraction are important. In a landscape with homogeneous basins, there is positive correlation between the value of a local optimum and the size of its basin, i.e., better local optima have bigger basins, which is a desirable characteristic [92]. Furthermore, if the basins have the same size, the landscape has a deterministic configuration (Fig. 4c); otherwise the landscape has a random configuration (Fig. 4d) [40].

3.3. Benchmark functions

The concepts of basins of attraction, neighborhood and fitness landscape are useful to describe the characteristics of a given optimization problem. A qualitative description – using labels such as rugged or unimodal – of a function can be made when its structure and characteristics are known. For example, Mersmann et al. describe in [92] the characteristics of 24 problems from the COmparison Continuous Optimizer (COCO) noiseless set [48].

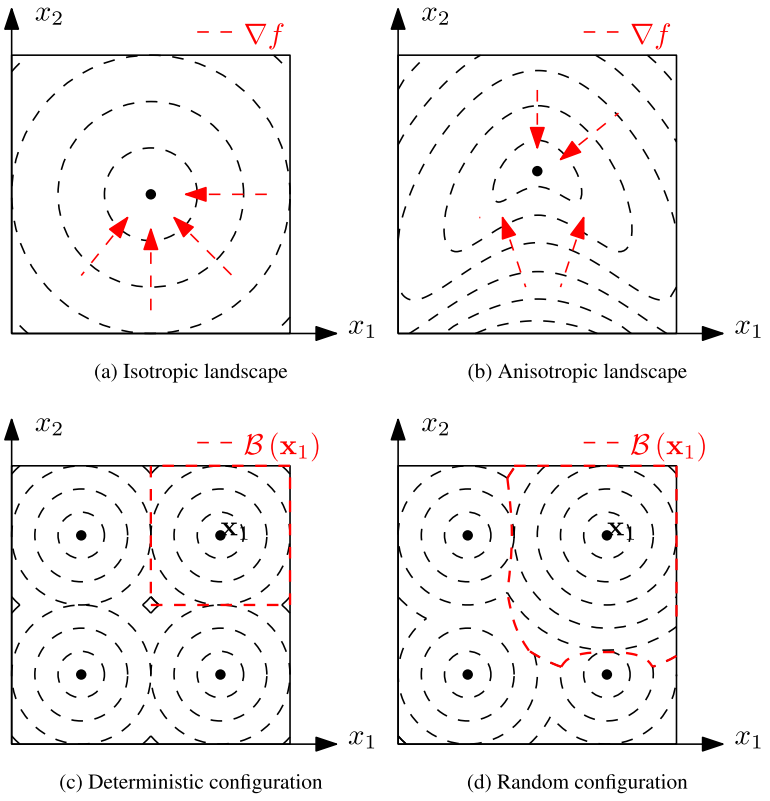


Fig. 4. Contour plots of four typical function in two dimensions, where each plot depicts a specific landscape characteristic. Each dark dashed line represents a fitness levelset. The red lines in (a) and (b) represent the gradient direction, ∇f. The red lines in (c) and (d) represent the size of the basin of attraction, $\mathcal{B}(\mathbf{x}_1)$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Benchmark functions, such as those in the COCO set, are employed to evaluate algorithm performance. Since their structure is known, empirical observations can be made regarding the characteristics that make a problem difficult for the algorithm under evaluation. The noiseless set in COCO is composed of 24 scalable functions, divided into five groups: separable, low or moderately conditioned, highly conditioned and unimodal, multimodal with adequate global structure, and multimodal with weak global structure. The noisy set is composed of eight basis functions, with different types and levels of noise [48]. Due to its use at the annual BBOB workshops series at the Genetic and Evolutionary Computation Conference, it has been widely used for comparing algorithms.[3] Similar workshops are carried out every year at the Congress on Evolutionary Computation. In 2005, the focus was on low dimensionality ($D \leq 50$) problems [146]. The benchmark set is composed of 25 test functions divided in four groups: unimodal, basic multimodal, expanded multimodal, and hybrid composition. In 2013, the focus was on high dimensionality ($D \geqslant 1000$) [79] or multimodality [78] problems. The high dimensional set is composed of 15 functions divided into four groups: fully separable, partially additively separable, overlapping, and fully non-separable. The multimodal set was composed of 20 functions.

In addition to evaluating algorithm performance, these benchmark sets are useful to build meta-models – as mentioned in Section 2. Furthermore, characteristics such as modality and smoothness are unknown for most real world problems. Since the only information available for BCOPs are the input and output samples, ELA measures are the only approach to acquire appropriate knowledge about the problem characteristics. Therefore, these benchmark sets are also useful to evaluate new ELA methods. We will discuss the ELA measures in detail in Section 6.

4. Algorithm space: Search algorithms for continuous optimization

The second component of the algorithm selection framework is the algorithm space, $\mathcal{A}$. It was defined in Section 2 as the smallest set of complementary and robust algorithms that solves all the problems in $\mathcal{F}$ with better than average performance. Complementary algorithms solve different types of problems. Robust algorithms have scientifically demonstrable accuracy and precision. All algorithms are iterative processes, i.e., they search for $\mathbf{x}_t$ by generating and improving candidate solutions. This is achieved by keeping a model of the problem, i.e., a set of assumptions about the relationship between input

[3] The raw data from these tests are available at http://coco.gforge.inria.fr/doku.php.

and output variables. The model defines how the algorithm gathers, processes and stores information at each iteration. Ideally, $\mathbf{x}_t$ is found in finite time. However, an algorithm may converge prematurely when it generates candidates in a small area of $\mathcal{X}$, none of which is $\mathbf{x}_t$. Premature convergence is a direct consequence of the model [62].

The hundreds, perhaps thousands, of algorithms reported in the literature can be broadly classified as deterministic or stochastic [21,129], depending on the model. Deterministic algorithms are based on rigorous formulations without random variables. Therefore, their results are unequivocal and replicable for the same initial conditions. Their model relies on linear algebra or computational geometry techniques. Most require the computation of a numerically stable estimator of the Gradient or the Hessian, i.e., the first or second derivatives, of the function. Deterministic algorithms quickly converge towards a local optimum [21]. It is common to re-start the algorithm at different, often random, locations to improve the convergence towards a global optimum [129]. Deterministic algorithms can be broadly classified into three groups: line, trust region and pattern search methods. On the other hand, stochastic algorithms rely on heuristics, statistical models and random variables to find a global optimum. Although advances have been made on their theoretical underpinnings [5], stochastic algorithms are less rigorously designed. However, they allow a more thorough exploration of $\mathcal{X}$. Compared with deterministic algorithms, they converge slowly towards the global optimum [22]. Stochastic algorithms can be broadly classified into random search methods, simulated annealing, and population based algorithms. Table 1 summarizes the main characteristics of each algorithm class.

Table 1
Main characteristics of the major algorithm families for black-box continuous optimization.

Deterministic algorithms	
Family:	Line search methods
Description:	These methods generate new candidates, $\mathbf{x}_{i+1}$, by searching along a direction $\Delta\mathbf{x}$, i.e., $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$ [21,129]. The simplest line search method is *gradient descent*, which uses the first derivative at the current candidate, $\mathbf{x}_i$, as direction, i.e., $\Delta\mathbf{x} = -\varrho\nabla f(\mathbf{x}_i)$, where ϱ is the step length. To improve convergence, the *conjugate gradient* method uses the current and previous candidates' gradients to calculate the direction [107]. In contrast, the *Newton* method uses the second derivative, i.e., $\Delta\mathbf{x} = -\varrho\left[\nabla^2 f(\mathbf{x}_i)\right]^{-1}\nabla f(\mathbf{x}_i)$, where $\nabla^2 f(\mathbf{x}_i)$ is the Hessian matrix. It is possible that $\nabla^2 f(\mathbf{x}_i)$ is expensive to calculate exactly, but an approximation is sufficient to generate a direction. These methods are known as *Quasi-Newton*. The most common are Broyden-Fletcher-Goldfarb-Shanno (BFGS), Davidon-Fletcher-Powell (DFP), and Symmetric Rank 1 (SR1) [109]
Family:	Trust region methods
Description:	These methods assume that the function in the neighborhood of $\mathbf{x}_i$ can be approximated by a linear or quadratic function, $g(\mathbf{x}_i)$, whose optimum correspond to $\mathbf{x}_{i+1}$, i.e., $\lvert\nabla g(\mathbf{x}_{i+1})\rvert = 0$. The ratio between the real change in fitness, $f(\mathbf{x}_{i+1}) - f(\mathbf{x}_i)$, over the estimated change in fitness, $g(\mathbf{x}_{i+1}) - g(\mathbf{x}_i)$, is compared against a threshold. If the ratio exceeds the threshold, $\mathbf{x}_{i+1}$ is accepted as the new solution and the neighborhood radius is decreased [110]. Otherwise, $\mathbf{x}_{i+1}$ is rejected and the neighborhood radius is increased [21,129]. Perhaps the earliest method of this type is the *Levenberg–Marquardt algorithm* [24]. Trust region methods are thought to be dual to line search methods. On the former, the step size is selected first – the trust region – followed by the direction. On the later, the direction is selected first followed by the step size [108]
Family:	Pattern search methods
Description:	These methods use a set of candidates around $\mathbf{x}_i$ to direct the search [129]. The simplest of these methods is the *coordinate search algorithm*. On each iteration, all the components of $\mathbf{x}_i$ are fixed except for one, which is updated until a fitter value is found. This process is carried out for all components in subsequent iterations. A more efficient method is the *Nelder-Mead algorithm* [21], which uses a simplex, i.e., the D dimensional analog of a triangle with $D+1$ vertices. The simplex is updated at each iteration by reflecting the vertex with the lowest fitness, $\mathbf{x}_{i+1}$. If the vertex continues to be less fit, the simplex is contracted by reducing the distance between $\mathbf{x}_{i+1}$ and $\mathbf{x}_i$. When the vertex improves, the simplex is expanded by increasing the distance between $\mathbf{x}_{i+1}$ and $\mathbf{x}_i$. If one of the vertices does not change over few iterations, its distance with $\mathbf{x}_i$ is decreased [21]
Stochastic algorithms	
Family:	Random search methods
Description:	These methods sample $\mathcal{X}$ using a fixed or adaptive probability distribution, usually normal or uniform [72,84]. Random search methods converge with probability one to the global optimum when the sample size converges to infinity [129], and are computationally more efficient than grid search [11]. An example of this type of methods is uninformed random picking [57]
Family:	Simulated annealing methods
Description:	These methods mimic the process in which a crystalline solid is heated and then allowed to slowly cool until it achieves its most regular possible crystal lattice configuration [55,22]. Simulated annealing methods always select candidates with improved fitness. However, they also allow the selection candidates with non-improving fitness; hence, providing means to escape local optima. The probability of selecting non-improving candidates depends on the *temperature* parameter, which decreases at each iteration of the algorithm [55]
Family:	Population based algorithms
Description:	These methods manipulate a group or "population" of candidate solutions simultaneously [58]. They are often inspired by a biological phenomenon; hence, they are often called bio-inspired computing. These methods can be broadly classified into evolutionary and swarm intelligence algorithms. Evolutionary algorithms use natural selection as a model of the optimization process. These algorithms modify the population at each iteration, first by selecting the fittest members, and then exchanging information between two or more candidates or by randomly modifying one or more candidates. Some popular methods are Genetic Algorithms [164], Evolutionary Strategies [12], Estimation of Distribution Algorithms [52], and Differential Evolution [27]. On the other hand, swarm intelligence algorithms emulate the collective behavior of self-organized and decentralized systems, e.g., ant colonies and fish schools. Some popular methods are Ant Colony Optimization [30], Particle Swarm Optimization [31], Artificial Bee Colony [71], Bacteria Foraging Algorithm [113], and Artificial Immune Systems [51]

The large algorithmic diversity also hides some unpleasant surprises, particularly in the stochastic algorithmic families. In recent years, a stream of papers were published presenting novel methods inspired by some natural phenomenon. Invariably, these papers demonstrate the superiority of the new method. Hence, they are followed with numerous follow up papers showing the application of the new method to different sets of problems, always with exceptionally good results [142]. However, it has been shown that several methods recycle well known ideas and their performance is overestimated [25,116,155,163]. Teasing out the novelty of such methods is an arduous task, which should be carried out by the algorithm authors [133]. However, it is often the case that their main area of research is not optimization but a specific application [142]. It seems that metaphors have shifted from inspiration to justification for new algorithms [142], but instead of advancing the state-of-the-art, it obfuscates more important innovations in the field.

To sum up, the algorithm space's diversity complicates the selection problem. Furthermore, the relationship between the problem characteristics and the algorithm model is vague. Hence, practitioners often select, modify, hybridize, and even propose algorithms hoping to achieve an acceptable performance level. Such an approach assumes that any algorithm perform efficiently on any relevant problem. However, this is not a valid assumption, as there is a theoretical set of algorithms, such that each one of them solves a large subset of problems in $\mathcal{F}$ with the best possible performance [6].

5. Performance space: measures of algorithm performance

The third component of the algorithm selection framework is the performance space, $\mathcal{P} \subset \mathbb{R}$, which is the set of feasible values of a measure of an algorithm's robustness – how often is a quality solution found – or its efficiency – how many resources are needed to find a quality solution [7]. A performance measure, $\rho(f, \alpha)$, should be simple, quantitative, well-defined, interpretable, reproducible, fair, and relevant to practice [7,48].

Fig. 5a illustrates both robustness and efficiency measurements, where the vertical lines represent fixed computational resources (the number of function evaluations) and the horizontal lines represent fixed solution qualities (fitness values). Robustness measures use fixed resources [7]. Hence, they are compatible with real world applications where resources are limited [48]. However, robustness measures may be hard to interpret as finding a m-times fitter solution is linked to the possible unknown problem difficulty [48]. On the other hand, efficiency measures use fixed solution qualities [7]. Hence, they are preferable for comparing algorithms. It is evident that an algorithm is superior if it reaches a target solution m-times faster than any other [48].

There are several performance measures reported in the literature [7]. However, the *expected running time*, $\hat{t}$, is the measure of choice for most benchmark comparisons [48,146]. An efficiency measure, $\hat{t}$ estimates the average number of function evaluations required by an algorithm to reach the target solution, y_t, for the first time [121]. It is calculated over a number of algorithm runs as follows:

$$\hat{t}(f, \alpha, y_t) = \frac{\#\mathrm{FEs}(y_{\mathrm{best}} \geqslant y_t)}{\#\mathrm{succ}} \tag{2}$$

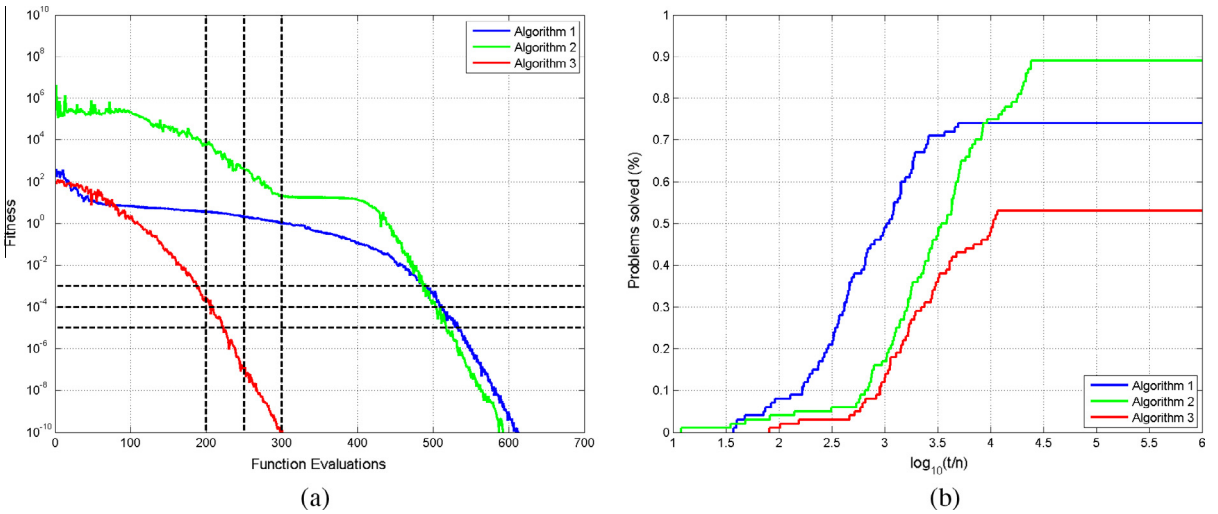


Fig. 5. Measurements of algorithm performance. (a) Presents a plot of the solution quality, measured as the current fitness, against the number of function evaluations for three algorithms in the same problem. The vertical lines indicate the limits on computational resources. The horizontal lines represent solution quality targets. At the cutoffs defined by these lines, the Algorithm 3 is the best performing while Algorithm 1 is the worst performing. (b) Shows the performance of the same three algorithms now over a set of problems. The performance is measured as the percentage of solved problems for a given number of log-normalized function evaluations, $\log_{10}(\hat{t}/D)$. A problem is solved if the algorithm reaches a solution quality target. The normalized number of evaluations is on the horizontal axis, whereas the percentage of solved problems is on the vertical axis.

where $\#\text{FEs}(y_{\text{best}} \geqslant y_t)$ is the number of function evaluations across all runs where the best fitness, y_{best}, is larger than the target, and #succ is the number of successful runs. With censored runs, $\hat{t}$ depends on the termination criteria. Additionally, $\hat{t}$ can measure performance across a set of problems. For example, Fig. 5b illustrates the probability of solving a problem for a given normalized budget, measured as $\log_{10} \hat{t}/D$. The probability plateaus when the problems cannot be solved. This representation identifies whether an algorithm is more efficient than another on average, and how likely it is that the algorithm will find a solution. In the figure, none of the algorithms solve all the problems. Algorithm 1 is the fastest of the three, but it solves over 70% of the problems. Algorithm 2 is computationally more expensive, however, it solves nearly 90% of the problems. The worst performing is Algorithm 3, as it is the most computationally expensive, and solves slightly above 50% of the problems.

The expected running time has two limitations: First, it requires a target, which may be unknown on a real world problem. However, y_t can be defined as the minimum acceptable improvement over the best known candidate. Second, it requires a sufficiently large number of algorithm runs to guarantee statistically significant results, which can be extremely time consuming depending on the problem [48]. The later limitation can be mitigated through bootstrapping [32].

Whichever measure is employed, performance is a random variable. It changes across problems, instances, even runs [47]. Furthermore, its probability distribution exhibits heavy-tailed behavior, which is ameliorated by restarting the algorithm from random positions [42]. By controlling this behavior, performance can be modeled as a parametric distribution that is totally described by few statistics, such as its mean and variance [62]. However, the performance distribution does not provide insights on its dependency with the problem characteristics.

6. Characteristics space: exploratory landscape analysis methods

The fourth component of the algorithm selection framework is the characteristics space, $\mathcal{C} \subset \mathbb{R}^m$, defined by the set of measures that provide information about the complexities of the problems, or the advantages and disadvantages of the algorithms. As mentioned in Section 3.2, data driven methods, know as ELA methods, are the only valid approach to measure the problem characteristics for BCOPs. ELA is an umbrella term for analytical, approximated and non-predictive methods [53,66,86,91,104] originally developed for combinatorial optimization problems [137]. For BCOPs, the existing methods are adaptations from their combinatorial counterparts [19,20,85,95,101,103,144,150], or purposely built for continuous spaces [19,83,91,98,120].

The number of reported ELA methods is meager compared with the number of reported search algorithms [86,95,118], due to the lack of clarity on the effects that the characteristics have on performance. Furthermore, the computational cost of using an ELA method may be greater than running a search algorithm [9,53,137]. Therefore, the focus should be on methods that are easily defined and computed, based on established statistical methods [9,59,126]. Besides, it is useful to classify these methods into types according to their focus as global or local [117], or as their sampling approach as unstructured or structured.

Global methods process the complete sample to generate the final measure, providing an overview of the landscape structure while concealing information about the change of fitness between neighboring candidates [117]. Local methods split the sample into groups of neighboring candidates. Each group is independently processed to produce a partial measure, which goes through further processing to obtain a final result. Therefore, local methods evaluate the changes of fitness within neighborhoods [117]. For example, Fig. 6 illustrates a multimodal function and a sample of eight candidates. Each candidate has a neighborhood of size two or three, e.g., the neighborhood of $\mathbf{x}_2$ is $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$. A global method might be the average fitness of the eight candidates, i.e., $c = \frac{1}{8}\sum_{i=1}^{8} f(\mathbf{x}_i)$. On the other hand, a local method might be the sum over all neighborhoods, of the average fitness over each neighborhood, i.e., $c = \frac{1}{2}\sum_{i=1}^{2} f(\mathbf{x}_i) + \frac{1}{3}\sum_{i=1}^{3} f(\mathbf{x}_i) + \cdots + \frac{1}{2}\sum_{i=7}^{8} f(\mathbf{x}_i)$.

Unstructured methods use a sample where each candidate can be considered an independent random variable. Structured methods use a sample where each candidate may be dependent on one or more previous candidates. For example, uniform sampling generates independent samples, while a random walk generates dependent samples. Furthermore, a dependent sample may also be biased, i.e., there may be areas of $\mathcal{X}$ more densely sampled than others. This implies that the samples used by either class are incompatible: the sample used by an unstructured method may not be suitable for a structured one and vice versa. Also, the sample used by a structured method may not be suitable for another structured

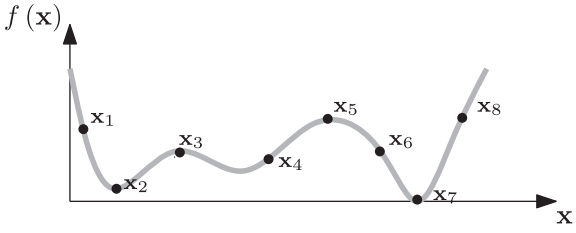


Fig. 6. A multimodal function and a sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_8\}$ of eight candidates. Each candidate has a neighborhood of size two or three.

method. This difference is fundamental, as unstructured methods allow us to calculate different measures, regardless if they are global or local.

Using these concepts, we classify the ELA methods in four types, as illustrated in Fig. 7:

- Type I methods are global and unstructured; hence, they can be applied to a *representative* sample of the input space. Type I methods are simple and scalable with D; however, they do not measure the probability of improving the fitness by sampling within a neighborhood.
- Type II methods are local and unstructured; hence, they can be applied to a representative sample. To define the neighborhood, it is necessary to provide a radius, r, which depends on the problem and the sample distribution. Therefore, selecting r is difficult. Additionally, calculating each neighborhood becomes intractable as the sample size increases.
- Type III methods are global and structured. Since the sample can be biased, it might not be possible to reuse it in other methods. Hence, the computational cost increases with the number of methods employed. A preprocessing method may remove the bias at an additional computational cost [101].
- [Type IV methods are local and structured; hence, they posses the limitations of Types II and III.

Although a different classification criteria could be followed (for example the use of distance metrics), this classification emphasizes the advantages of one type over the others. Arguably, combinations of Type I and II methods are preferable as the computational cost is reduced by sampling once, which is advantageous with expensive sampling, e.g., when the problem involves a real time process. Even if sampling is cheap, the computational cost of Type II, III or IV methods is not justifiable, as it may be faster to run several algorithms in parallel. Table 2 summarizes the methods discussed in the following sections, classified into types.

6.1. Type I: Global unstructured methods

A first set of type I methods assume that smooth landscapes have neighboring candidates with similar fitness; therefore, these measures are indicators of the landscape modality and the global structure strength. Fitness distance correlation, *FDC*, and Dispersion, $DISP_{100\varepsilon\%}$, measure the relationship between the candidate's location and its fitness [68,83]. *FDC* is calculated using the Pearson correlation between d, which is the Euclidean distance to the fittest candidate from the sample $\mathbf{X}$ of size n [95], and y. On the other hand, $DISP_{100\varepsilon\%}$ is the average distance between the $\epsilon = \lfloor \varepsilon n \rceil$ lowest cost candidates from $\mathbf{X}$, where $\varepsilon \in [0, 1]$. $DISP_{100\varepsilon\%}$ is normalized over the diagonal of $\mathcal{X}$. $DISP_{100\varepsilon\%}$ has the highest discrimination power when $\varepsilon \to 0$ and $D < 10$ [99]. Both *FDC* and $DISP_{100\varepsilon\%}$ are invariant to translational shifts and orthogonal rotations on $\mathcal{X}$, which are *global isometries* of the Euclidean space and do not affect d. *FDC* has been applied in other domains besides BCOPs [10,95,149,150,152], and $DISP_{100\varepsilon\%}$ demonstrated why the CMA-ES algorithm is ineffective in multi-funnel, multimodal problems [83].

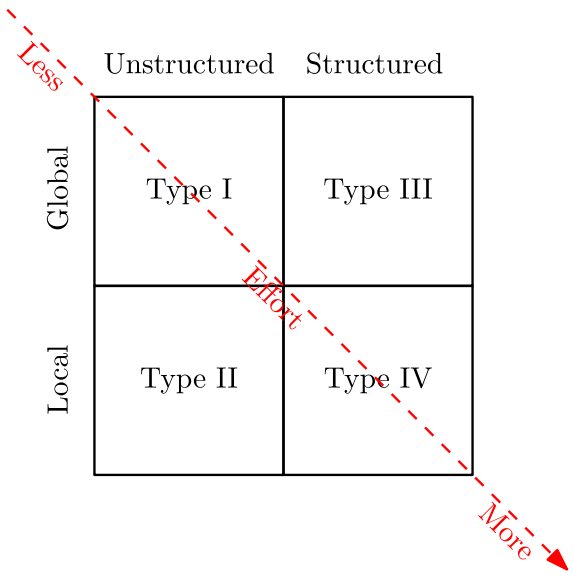


Fig. 7. Classification of the ELA methods into four types depending on the concepts of *order* and *neighborhood*. The arrow indicates the direction that the *computational effort* increments, and how this increment depends on the type.

Table 2
Summary of the ELA measures employed in this paper.

Type	Method	Measure	Description
I	Fitness distance correlation	FDC	Fitness distance correlation
	Dispersion	$DISP_{100\varepsilon\%}$	Dispersion of level ε
	Probability density function	$\gamma(\mathbf{Y})$	Skewness
		$\kappa(\mathbf{Y})$	Kurtosis
		$H(\mathbf{Y})$	Entropy
	Surrogate modeling	$\overline{R}^2_L$	Adjusted coefficient of determination of a linear regression model
		$\overline{R}^2_{LI}$	Adjusted coefficient of determination of a linear regression model plus interactions
		$\overline{R}^2_Q$	Adjusted coefficient of determination of a purely quadratic regression model
		$\overline{R}^2_{QI}$	Adjusted coefficient of determination of a quadratic regression model plus interactions
		$\min(\beta_L)$	Minimum of the absolute value of the linear model coefficients
		$\max(\beta_L)$	Maximum of the absolute value of the linear model coefficients
		CN	Ratio between the minimum and maximum absolute values of the quadratic term coefficients in the purely quadratic model
			Cross-validated classification error of the level-set model
	Information significance	$\xi^{(D)}$	Significance of order k
		$\varepsilon^{(k)}$	Entropic epistasis of order k
	Length scale	$pr(\Delta y/\|\Delta\mathbf{x}\|)$	Probability density function of $\Delta y/\|\Delta\mathbf{x}\|$
II	Fitness sequences	$H_{\max}$	Maximum information content
		ϵ_S	Settling sensitivity
		M_0	Initial partial information
	Evolvability	E_a	Evolvability
		P_i	Average escape probability
		NSC	Negative slope coefficient
		AEP	Accumulated escape probability
III	Probability of convexity		Ratio between the number of solution pairs for which the difference between their estimated and real cost is less than a threshold, and the total number of pairs
IV	Fitness sequences	ACF_τ	Auto-correlation function
		ℓ	Correlation length
		$\overline{ACF}$	Average auto-correlation
	Basin sizes and distribution	$\bar{\beta}$	Average basin size
			Average distance between local optima
			Size of the largest and fittest basins
			Distance between the largest and fittest basins
	Curvature	$pr(\|\nabla f\|)$	Probability density function of the gradient norm
			Probability density function of the Hessian condition number

A second set of methods measure the landscape modality and the global structure strength using an estimation of the probability density function (pdf) of $\mathcal{Y}$ [16,130]. For a piecewise invertible function, the pdf of $\mathcal{Y}$ is related to the first derivative by the Perron-Frobenius operator:

$$\mathrm{pdf}(y) = \sum_{x \in f^{-1}(y)} \frac{\mathrm{pdf}(x)}{|f'(x)|} \tag{3}$$

This implies that by characterizing the pdf of $\mathcal{Y}$ through its skewness, $\gamma(\mathbf{Y})$, kurtosis, $\kappa(\mathbf{Y})$, number of peaks, and entropy, $H(\mathbf{Y})$ [90,91], we obtain some information about the magnitude of the gradient of f.

A similar method characterizes the pdf of the difference in fitness, $y_i - y_j$, over the difference in position, $\|\mathbf{x}_i - \mathbf{x}_j\|$ [98]. The pdf is summarized using the entropy. The authors suggest sampling using a Lèvi flight, as it produces clusters of candidates uniformly distributed over $\mathcal{X}$; hence, it is thought to provide good coverage of $\mathcal{X}$.

A third set of methods use the fit of a regression or classification model as a measure of the landscape modality and global structure strength. The fit of a linear or quadratic regression model can be thought of as the distance to a reference problem [45,91], and it is evaluated using the adjusted coefficient of determination, $\overline{R}^2$. Moreover, variable scaling is measured using the maximum and minimum of the absolute value of the coefficients from the linear model without interactions, and the ratio between the minimum and the maximum absolute values of the quadratic term coefficients of a quadratic model without interactions [91].

For a classification model, the output sample is divided into two classes using a threshold, e.g., the lower or upper quartiles of the fitness distribution. According to [91], an unimodal function could be cleanly cut by the hyperplane defined by a linear or quadratic classifier, whereas a multimodal function could only be cleanly cut by a non-linear classifier. The fit is evaluated using the cross-validated miss-classification rate [91].

A fourth set of methods focuses on variable dependencies, i.e., the ease in which f can be broken down into simpler problems of lower D [92]. Most methods focus on linear interactions [28,37,105,124,127,128], e.g., the fit of a linear model with

interaction terms [91]. However, non-linear interactions may be expressed as the joint probability of a subset of variables, and measured using mutual information [132]. Let $\mathcal{V} = \{1, \ldots, D\}$ be a set of variable indexes, where $v \in \mathcal{V}$ is the index of a variable, and $V \subset \mathcal{V}$ is a combination of variables. The significance of $V, \xi(V)$, is the ratio between the mutual information, $I(\mathbf{X}_V; \mathbf{Y}) = H(\mathbf{X}_V) + H(\mathbf{Y}) - H(\mathbf{X}_V, \mathbf{Y})$, and the cost entropy, $H(\mathbf{Y})$. This ratio is also known as the uncertainty coefficient or Theil's U. The epistasis of a combination $V, \varepsilon(V)$, is calculated as follows:

$$\varepsilon(V) = \frac{I(\mathbf{X}_V; \mathbf{Y}) - \sum_{v \in V} I(\mathbf{X}_v; \mathbf{Y})}{I(\mathbf{X}_V; \mathbf{Y})} \tag{4}$$

The results are summarized using the average significance, $\xi^{(k)}$, and average epistasis, $\varepsilon^{(k)}$, of order k, where $k = |V|$.

6.2. Type II: Local unstructured methods

The first type II method analyzes a sequence of fitness values, $\mathbf{S} = \{y_1, \ldots, y_n\}$, obtained by sorting a uniformly distributed sample [103]. The starting element in the sequence is a candidate from the sample selected at random, while the remaining elements in the sequence are selected using the nearest neighbor heuristic, i.e., the element whose Euclidean distance is the lowest to the current element. To avoid backtracking, any candidate already in the sequence is excluded from the nearest neighbor calculations. Known as ICOFIS, it is an adaptation of the method described in [85,144,153], where a random walk is used to collect the sample and generate the sequence, instead of sorting a uniformly distributed random sample. However, this approach biases the sample unless $n \to \infty$ [103]. A bias correction method, such as stratified or weighted sampling [26,101,170], is needed when a random walk with $n \ll \infty$ is used.

Let $\Phi(\epsilon) = \{\phi_1, \ldots, \phi_{n-1}\}$ be a symbol sequence where $\phi_i \in \{\bar{1}, 0, 1\}$, converted from $\mathbf{S}$ by following the rule:

$$\Psi(i, \epsilon) = \begin{cases} \bar{1} & \text{if } \frac{\Delta y}{\|\Delta \mathbf{x}\|} < -\epsilon \\ 0 & \text{if } \left|\frac{\Delta y}{\|\Delta \mathbf{x}\|}\right| \leqslant \epsilon \\ 1 & \text{if } \frac{\Delta y}{\|\Delta \mathbf{x}\|} > \epsilon \end{cases} \tag{5}$$

where Δy is the difference between y_{i+1} and y_i, $\|\Delta \mathbf{x}\|$ is the Euclidean distance between $\mathbf{x}_{i+1}$ and $\mathbf{x}_i$, and $\epsilon \geqslant 0$ is a sensitivity parameter that sets the accuracy of $\Phi(\epsilon)$. For example, $\Phi(0)$ has a zero if and only if there are neutral areas in the landscape. On the other hand, $\Phi(\epsilon)$ is all zeros if ϵ is larger than the maximum Δy. The method accounts for the uncertainty added by the step size represented by $\|\Delta \mathbf{x}\|$. The value of $\frac{\Delta y}{\|\Delta \mathbf{x}\|}$ converges to the derivative if $\|\Delta \mathbf{x}\| \to 0$. Two consecutive symbols compose a block, which represents a slope, peak or neutral area in the landscape. The information content of $\mathbf{S}$ is defined as $H(\epsilon) = -\sum_{a \neq b} p_{ab} \log_6 p_{ab}$, where $a, b \in \{\bar{1}, 0, 1\}$ and p_{ab} is the probability of finding the block ab in the symbol sequence. The logarithm base is six because this is the number of possible blocks where $a \neq b$. $H(\epsilon)$ is bound between $[0, 1]$, and $0 \log_6 0 \equiv 0$.

The result from $H(\epsilon)$ is not an explicit measure of the landscape smoothness [153]. For this purpose, a new sequence, $\Phi'(\epsilon)$, is constructed from $\Phi(\epsilon)$ by removing all the zeros and repeated symbols. $\Phi'(\epsilon)$ has the form "$\ldots \bar{1}1\bar{1}1\bar{1} \ldots$" and represents the changes in concavity encountered during $\mathbf{S}$. The partial information content, $M(\epsilon)$, characterizes the landscape smoothness [153] and it is equal to $M(\epsilon) = \frac{|\Phi'|}{n-1}$ with $n \gg 1$. The results are plotted against ϵ resulting in the curves in Fig. 8, which are summarized by the following measures: Maximum information content, $H_{\max} = \max_\epsilon \{H(\epsilon)\}$, settling sensitivity, $\epsilon_S = \log_{10}(\min_\epsilon \{\epsilon : H(\epsilon) < 0.05\})$, initial partial information, $M_0 = M(\epsilon = 0)$. The three measures correspond to points on the $H(\epsilon)$ and $M(\epsilon)$ curves as illustrated on Fig. 8. At H_{max}, $\Phi(\epsilon)$ has the highest diversity. Hence, rugged landscapes are expected to have high value of H_{max} [85]. At ϵ_S, $\Phi(\epsilon)$ is nearly all zeros. Hence, it represents the maximum change of fitness found during $\mathbf{S}$ and indicates the scaling of the problem. Furthermore, ϵ_S is strongly correlated (≈ 0.96) to the entropy of the fitness pdf, $H(\mathbf{Y})$ [103]. At M_0, $\Phi(\epsilon)$ has the highest number of inflexion points, providing information about the landscape ruggedness. The accuracy of the measures increases with the number of distinct values of ϵ used.

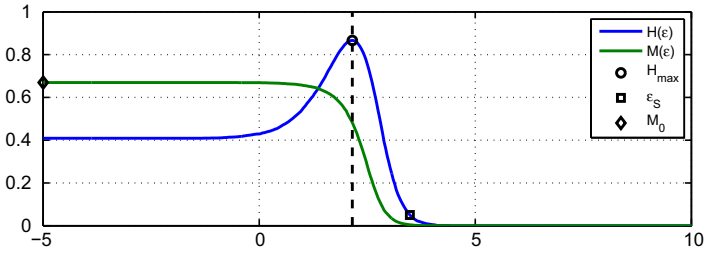


Fig. 8. Typical curves for $H(\epsilon)$ and $M(\epsilon)$ and their derived measures. The horizontal axis represents the sensitivity parameter ϵ on a $\log_{10}$ scale. The black dashed lines are visualization aids.

A second set of measures is based on the estimated escape probability, or evolvability, of a fitness landscape, E_a, which is loosely defined as the ability of an individual or population to generate fitter candidates [135]. More formally, evolvability is defined as the probability that a fitter candidate can be found within a candidate's neighborhood [135] as follows:

$$E_a(y_i) \approx \frac{|\{y : y \in \mathcal{N}_r(y_i), y_i \geqslant y\}|}{|\mathcal{N}_r(y_i)|} \tag{6}$$

Therefore, high evolvability means that there is a high probability of finding a fitter candidate within the neighborhood. Hence, a search algorithm is likely to be successful in exploring the landscape. On the other hand, low evolvability means that it is unlikely that fitter candidates will be found within the neighborhood. Therefore, a search algorithm will find difficult to explore the landscape efficiently. There are three methods that use the concept of evolvability to analyze a landscape: The evolvability portrait, which is a plot of E_a against y_i [135]; the fitness–fitness cloud, which is a plot of the fitness of the neighbors against y_i [151]; or a fitness–probability cloud, which is a plot of the average escape probability, P_i, against y_i [82], with P_i calculated as follows:

$$P_i = \frac{\sum_{y_j \in C_i} E_a(y_j)}{|C_i|} \tag{7}$$

where $C_i = \{y | y \geqslant y_i\}$, i.e., P_i is the average evolvability of all the candidates with worse fitness than the current candidate. Ideally, a smooth trend should be observed on these plots from the unfit to the fitter candidates. The trend may be summarized by the negative slope coefficient [151], *NCS*, for the fitness–fitness cloud, and the accumulated escape probability [82], *AEP*, for the fitness–probability cloud. Although both measures are simple to calculate, they are impractical due to their high computational cost and dependency to the neighborhood radius, r.

6.3. Type III: Global structured methods

Type III methods are perhaps the less common type reported in the literature. An example of these methods measures the probability that the space between two candidates is convex [91]. A candidate $\mathbf{x}_k$ is generated using a linear combination of two other candidates, $\{\mathbf{x}_i, \mathbf{x}_j\}$, and its fitness, $\hat{y}_k$, is estimated using a convex combination of $\{y_i, y_j\}$. The probability of convexity is defined as the ratio between the number of $\{\mathbf{x}_i, \mathbf{x}_j\}$ combinations for which the difference between y_k and $\hat{y}_k$ is less than a threshold, and the total number of $\{\mathbf{x}_i, \mathbf{x}_j\}$ combinations.

6.4. Type IV: Local structured methods

A first set of type IV methods analyze a sequence of fitness values, $\mathbf{S}$, obtained from a random walk over $\mathcal{X}$. This includes the original ICOFIS [85,144,153], the structured version of the type II method presented in Section 6.2. The auto-correlation function of $\mathbf{S}$, ACF_τ, is one of the earliest proposed ELA methods [160]. It is calculated as follows [93]:

$$ACF_\tau = \frac{1}{\hat{\sigma}_y^2 (n-\tau)} \sum_{i=1}^{n-\tau} (y_i - \bar{y})(y_{i+\tau} - \bar{y}) \tag{8}$$

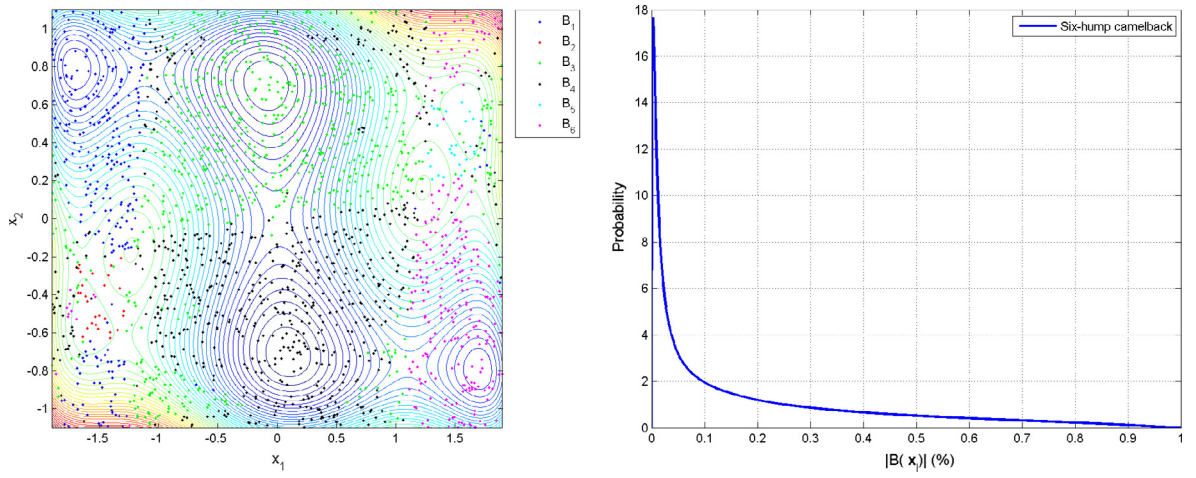
where τ is the number of delays over which the auto-correlation is calculated, $\bar{y}$ is the mean fitness, and $\hat{\sigma}_y^2$ is the sample fitness variance. It has been suggested that ACF_1, also known as correlation length or nearest neighbor auto-correlation, captures the landscape smoothness efficiently [143]. Assuming that ACF_τ is an exponential function of τ, the normalized correlation length, ℓ, is zero if ACF_1 is zero. Otherwise, $\ell = -\ln(|ACF_1|)^{-1}$. A high value of ℓ implies that a local search finds a high fitness solution after a high number of function evaluations [93], whereas a low value of ℓ implies that the algorithm converges prematurely on low fitness solutions. It was thought that ACF_τ, plus the mean and variance of the fitness pdf, would completely characterize a landscape [161]. However, it has been demonstrated neither *ACF* nor ℓ are sufficient to characterize all landscapes [118].

For continuous optimization problems, *ACF* has been defined as [90]:

$$ACF(r) = \frac{1}{\sigma_y^2} \frac{1}{|\mathcal{N}_r|} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{N}_r} (y_i y_j - (y_i + y_j)\bar{y} + \bar{y}^2), \tag{9}$$

where $\mathcal{N}_r(\mathbf{x}) = \{\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\| = r\}$. Over a set of values of $r, \mathcal{R}$, the average auto-correlation is defined as the sum of $ACF(r)$ over the cardinality of $\mathcal{R}$. With a uniformly distributed random sample, it is unlikely that $|\mathcal{N}_r(\mathbf{x}_i)| \gg 1$ for any r. Hence, candidates are sampled in pairs. The first one is selected from anywhere in $\mathcal{X}$, while the second is selected at a distance of $r \in \mathcal{R}$ from the first.

A second set of type IV methods estimate the pdf of the size of the basins of attraction, $|\mathcal{B}(\mathbf{x}_l)|$, defined as the volume from $\mathcal{X}$ occupied by each basin [40]. As described in Section 3.2, to estimate the size of a basin, a local search starts from a random candidate in $\mathcal{X}$, stopping only when it converges to a local optimum. After carrying out n local searches, a basin is observed if at least one search has converged to it. Fig. 9a illustrates the results for the Six-hump Camel-back function. The function is



(a) (b) Probability distribution of basin sizes

Fig. 9. Basins of attraction for the Six-hump Camel-back function estimated using a local search method [40]. (a) Is a contour plot, where each mark represents one of 2000 candidates used as starting points for the local searches, their color representing one of six observed basins. (b) Is the best-fitting gamma distribution of the basin volume, measured as a percentage of $\mathcal{X}$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

presented a contour plot, where each mark represents one of 2000 candidates used as starting points for the local searches, their color representing one of six observed basins.

The volume of an observed basin could be estimated as the percentage of candidates that where drawn to its local optima. However, this ignores the volume of potentially unobserved basins. For this purpose, a gamma [40] or exponential [17] pdf is fitted to the data, as illustrated in Fig. 9b for the Six-hump Camel-back function. In the figure, the horizontal axis represents the volume of the basin measured as a percentage of $\mathcal{X}$. Let $\bar{\beta}$ be the average basin volume, which is estimated as follows:

$$\bar{\beta} = \frac{1 - \left(1 + \frac{\vartheta}{\gamma_0}\right)^{\gamma_0}}{\alpha} \tag{10}$$

where γ_0 is the parameter from the best fitting gamma distribution. According to [40], Eq. (10) has a solution at ϑ equal to the ratio between the sample size and the total number of basins. To improve the estimation, non-parametric method may be used for large input spaces [123]. Alternatively, a random walk may be used instead of the local search [3]. Furthermore, the size, number and location of the local optima are used to calculate additional measures: the average distance between local optima [19,158], and the size and distance between the largest and fittest basins [19].

These methods are expensive, as they require numerous function evaluations. For example, for the 2000 candidates illustrated on Fig. 9, nearly 2.6×10^5 function evaluations were used in the local searches. Alternatively, only the starting points may be used to estimate the number of basins [101]. Using a Delaunay triangulation, a network is constructed in which the local search is carried out. Although the cost in function evaluations decreases, calculating the Delaunay triangulation is computationally expensive for $D > 3$, and limited to $D \leq 8$.

A third set of type IV methods measures the function curvature by numerically estimating the Gradient and Hessian at each candidate, using extrapolation [91] or finite differences. The results are summarized using the pdf of the Euclidean norm of the gradient and the condition number of the Hessian.

6.5. Discussion

In summary, ELA methods attempt to quantify the landscape complexity, focusing on the characteristics described on Section 3. Some methods use a specific sampling procedure. Others require information about the candidates' neighborhood. Hence, ELA methods range from computationally simple (type I) to complex (type IV).

ELA methods are not exempt from criticisms. First, most of the work attempts to provide a single, all encompassing, measure of complexity. This approach is optimistic, as it is the interplay between characteristics that defines difficulty. As such, several complementary measures are necessary [10,95,135]. Second, ELA methods require a large sample to be precise [66,95,149]. The sample size grows exponentially with D; hence, ELA methods are imprecise in polynomial time [53]. However, as ELA methods are rooted in statistical analysis, their uncertainty can be estimated using resampling methods. Fewer function evaluations may be needed if the methods are used during an algorithm run [104]. However, the bias imposed by the algorithm must be corrected to avoid deceiving results [101].

Third, it is unclear whether the existing methods are sufficient, even necessary. Since the difficulty of a problem is relative to the algorithm used [54], an ELA method lacks utility if it fails to provide information about the strengths and weaknesses of any algorithm. However, there is evidence that even type I methods are sufficient to identify a good algorithm for a given optimization problem [1,14,91,102,103]. We discuss these contributions in the following section.

7. The selection framework and related techniques for black-box continuous optimization problems

The algorithm selection framework based on the work of Rice [125] has been applied in many domains, including combinatorial auctions [39,74,75], clustering [2,77,76,148], feature selection [157], graph coloring [138,141], mixed integer programming [65,168], planning [61], program induction [44,45], quadratic assignment [136], satisfiability [39,62,65,69,87–89,166], scheduling [9,139], time series [46], the traveling salesman problem [65,70], among other domains [73,137]. Most of these works use one of two alternative implementations of a meta-model, both illustrated in Fig. 10 using a set of four algorithms as example. The first uses a classification model [125]. The second uses a regression model per algorithm to learn the map $g_i : \mathcal{C} \to \mathcal{P}$ [75]. For both approaches, it is required a knowledge base containing the characteristics of all problems in a subset, $F \subset \mathcal{F}$, and the performance of all algorithms in a subset, $A \subset \mathcal{A}$, for all problems in F. Both models are trained using the characteristics as input patterns. The classification model uses the best performing algorithm as the output pattern. The regression models use the performance measure as the output pattern, and a sorting method selects the algorithm with the best predicted performance.

There are trade-offs associated with each implementation. The classification model is monolithic; hence, it has fewer elements prone to failure than the regression model. However, adding or removing algorithms imply the re-training the classification model, whereas the regression model is modular. Another trade-off is a consequence of an algorithm failing to produce a solution a problem in F within the computational budget. Therefore, the algorithm run has been *censored*. The results from these runs leave gaps in the knowledge base, affecting the data quantity for regression models. To fill in the gap, the budget can be used as the real-run time, or as a lower bound used to estimate the real run-time [131]. However, censoring is not an issue for a classification model, if there is at least one algorithm that produces a solution for every problem in F.

In the BCOP domain, there is a paucity of work examining the algorithm selection problem or describing applications of Rice's framework [125]. For example, Francois and Lavergne [38] used a regression model to predict the performance of an

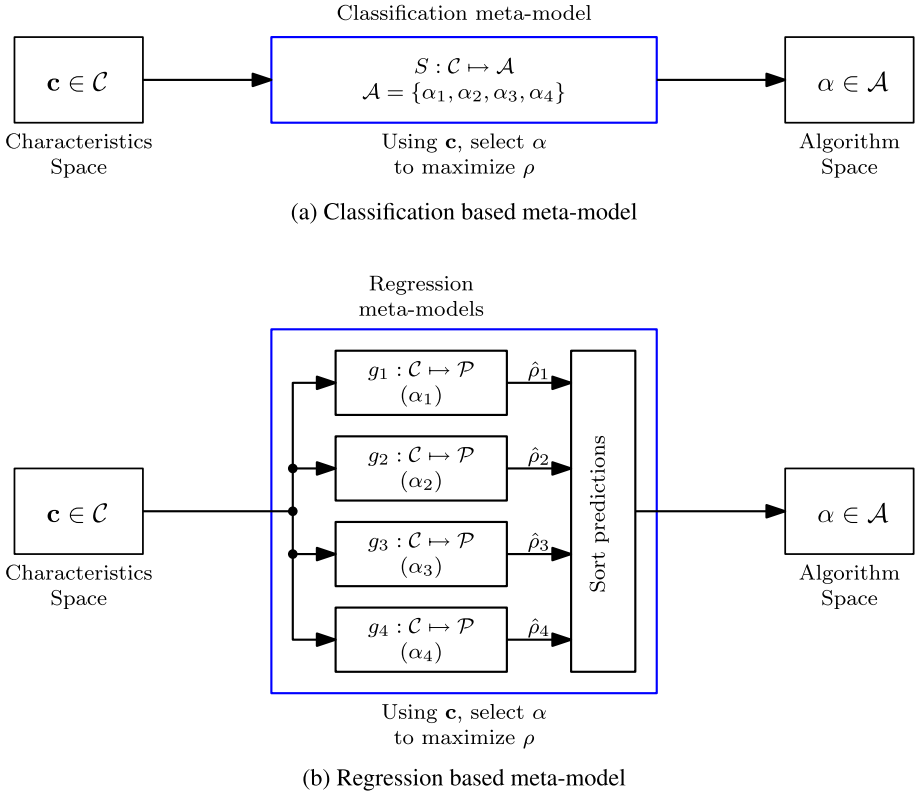


Fig. 10. Diagram of two approaches to construct a meta-model using experimental data and machine learning techniques. The models relate two components of the algorithm selection framework illustrated in Fig. 1: the characteristics and algorithm spaces.

evolutionary algorithm. The inputs to this model were limited to the algorithm's parameters. Although Francois and Lavergne hinted that problem classes could be related to performance, they did not provide any detail into how to determine such classes. Therefore, a new model had to be trained for each problem, limiting its application in realistic scenarios.

More recently, Bischl et al. [14] used a cost sensitive learning model to select the best algorithm between BFGS, BIPOP-CMA-ES, LSstep, and LSfminbnd. This type of models is trained to produce accurate performance predictions for the best algorithm only. The knowledge base was composed of 360 problem instances, drawn from the *Comparing Continuous Optimization* (COCO) benchmark set [49] at ten dimensions, which was deemed as a representative dimensionality. Each problem was characterized using 19 measurements, extracted using a mixture of type I, III and IV ELA methods. Additionally, the set of four algorithms was manually selected using the complete knowledge base. The model was verified using two cross-validation approaches. In the first one, five instances from each problem were used as test set; hence, the accuracy on unobserved instances of observed problems is estimated. On the second approach, all the instances from one problem were used as test set; hence, the accuracy on unobserved problems is estimated. However, their set of four algorithms was manually selected using the complete knowledge base, weakening the validation on unobserved problems. Furthermore, the results may not be generalizable for a knowledge base with problems of different dimensionalities.

In previous work [102], we have used a regression model to identify the best out of eight parameter combinations for the CMA-ES algorithm. The knowledge base was composed of 1800 problem instances, drawn from the COCO benchmark set at $\{2, 3, 5, 10, 20\}$ dimensions. Each problem was characterized using seven type I measurements and four parameters. Among the predictors was the target precision, included to minimize the gaps in the database due to censoring. However, this approach resulted in a model that often underestimated the performance on the hardest problems. Furthermore, with a sample size of $n \approx 10^5 \times D$, the ELA measurements were too expensive to be practical. During validation, we compared the model's accuracy against randomly selecting a configuration on unobserved instances of observed problems only.

Abell et al. [1] employed ISAC, a configuration method for portfolios of SAT solvers [69], to select the best performing algorithm from a group of 21 algorithms for the COCO benchmark set. The knowledge base was composed of 1289 problem instances at $\{2, 3, 5, 10, 20, 40\}$ dimensions, which excluded all instances with censored runs. The core of ISAC is a classification model, which uses ten characteristics as predictors, seven of which are extracted using type I and IV ELA methods. Therefore, the best performing selector required nearly 50 times more function evaluations than the baseline algorithm. Furthermore, the validation was limited to unobserved instances of observed problems.

These works demonstrate that the weakest link on the framework's application is the computational cost of the ELA methods. However, the selection framework is one of several related approaches, which aim to improve the performance by combining algorithms with different strengths. Perhaps, the simplest one is to run sequentially or in parallel a set of algorithms [41]. These *algorithm portfolios* aim to improve the performance by distributing the available resources among multiple algorithms as efficiently as possible [115]. For example, Montez de Oca et al. [96] identified and combined three fast and reliable PSO variants. Vrugt et al. [156] interleaved five meta-heuristics – CMA-ES, GA, PSO, DE and PCX. Peng et al. [115] distributed the function evaluation budget between the algorithms, and used a migration scheme to exchange information between algorithms. Statistical tests were used to identify and stop prematurely convergent algorithms, whose function evaluation budget was redistributed over the remaining ones.

Algorithm portfolios are closely related to hybrid meta-heuristics [15,154] and hyper-heuristics [47,112]. In the former, one or more algorithms are interleaved with the aim to produce synergies between them [15,154,156]. For example, a memetic algorithm pairs a population-based search method, usually an evolutionary algorithm, with a local refinement method, often a line search [23]. Hyper-heuristics construct algorithms by adding components to an "empty" one, or by improving iteratively a randomly generated initial algorithm [47,112]. The hyper-heuristic may be constructed on-line, i.e., while the problem is being solved, or off-line, i.e., the best performing hyper-heuristic over a set of problem instances is selected from a group [112]. Statistical racing can be thought of as an off-line hyper-heuristic. This method evaluates a set of candidate algorithms on a stream of instances, where an algorithm drops out from the race when sufficient evidence is collected against it [13].

Parameter tuning and control refers to the methods employed to automatically adjust the algorithm parameters [33,35]. Tuning aims to find a parameter set applicable to a wide range of optimization problems. Hence, it can also be thought of as an off-line hyper-heuristic. Tuning involves experimenting with different parameter sets over a suite of test problems, and selecting the best performing. Tuning has some drawbacks [29,33,34,114]: First, the parameters are mutually dependent and systematically testing all combinations is impractical. Second, even if the tuning effort was significant, the resulting parameters are not necessarily optimal for all problems. Third, tuning ignores the fact that an algorithm run is a dynamic and adaptive process. Fourth, it is unclear whether perceived similarity between problems implies similar optimal parameter set. Fifth, when a theory-based tunning approach is used, the complexities of the search and characteristics of the problems must be notably simplified.

On the other hand, control refers to the methods for adjusting the parameters during the run, potentially improving them while solving the problem [33]. These methods leverage the information accumulated about the problem during the search. Hence, they can be thought of as an on-line hyper-heuristic. For effective parameter control, only the most influential parameters should be adjusted. Arguably, control methods are a good example of over-engineering an already sophisticated adaptive system [29], or that control techniques introduce new selectable parameters to the algorithm [114]. Furthermore, control methods are against the principle of self-organization [134].

Algorithm portfolios, hybrid meta-heuristics, and hyper-heuristics have common disadvantages. They neglect the information collected on previous experiments, and they do not examine the similarity between the current problem with others previously observed [23]. Therefore, they fail to provide deeper insight into the relationship between problem structure and algorithms. As such, they are themselves black-boxes treated with suspicion by the users, who distrust anything that gives solutions without justifications [94]. Furthermore, they are subject to the NFLT [165], i.e., a method might work for some problems but fail in others [15]. Hence, the process of designing, selecting and adjusting search algorithms for BCOPs is cumbersome, requiring expert knowledge on several algorithms, and skills in algorithm engineering and statistics [15]. In other words, it remains an art rather than a science [97].

8. Conclusions and further research avenues

The problem of algorithm selection, that is identifying the most efficient algorithm for a given computational problem, is a non-trivial task. In this paper, we have presented a detailed review of key concepts, methods and evaluation techniques for algorithm selection for BCOPs. The algorithm selection framework proposed by Rice [125] was described in detail. This was followed by a description of the four components – problem, algorithm, performance and characteristic – couched in terms of the requirements for continuous optimization problems. Next, we proposed a classification of ELA methods based on computational costs and their focus on global or local information, and the sampling technique employed (unstructured and structured). Then, we discussed applications of the framework in the BCOP domain. Finally, the relationship between the algorithm selection framework and algorithm portfolios, hybrid meta-heuristics, and hyper-heuristics was also discussed. In the remainder of this section, we identify remaining challenges and propose future research directions.

The algorithm selection problem is ill-defined due to the complexity and size of the problem and algorithm spaces. Hence, a formal solution to the algorithm selection problem may not exist. Similar ill-defined problems are solved using decision support systems (DSS), i.e., computational systems that leverage data and models. The selection framework fits into this description, and it would bridge the gap between the collection and usage of experimental data [56] existing on related approaches such as hyper-heuristics. Furthermore, the framework may identify relative strengths and weaknesses of the algorithms [140]. Besides the limitations identified on [1,14,102], the BCOP domain adds challenges to the implementation of the DSS, most of them absent in other domains.

It is unclear, particularly for stochastic algorithms, how their performance is affected by the problem characteristics. Although theoretical analysis of stochastic algorithms has advanced significantly in the latest few years [5,67,106], it is still limited to simplified problems [111]. Nevertheless, theoretical insights clarify the effects of the problem characteristics, which in turn would focus the development of ELA methods.

Furthermore, there is limited, if any, information about the problem instance available beforehand in a BCOP. Gathering sufficient and accurate information through ELA methods requires numerous function evaluations, on top of the ones required by the search algorithm. Hence, performance of the DSS is affected by the accuracy of the characteristic measures in two ways. Assuming that accurate measures reduce the selection error of the DSS, we could collect a large sample to calculate the measures as accurately as possible, leaving a small proportion of the function evaluation budget for the search. On the other hand, assuming that more than one algorithm may solve the problem, we could allow a higher error on the measures by collecting a smaller sample, leaving a larger proportion of the budget for the search. The balance between measure accuracy and search budget is equivalent to the exploration/exploitation balance.

Ideally, the cost of calculating the measures should be encapsulated within the search budget. This may be achieved by restarting the algorithm from the sample instead of random positions, as described in Section 5, improving simultaneously the heavy-tailed behavior of the performance [42]. Alternatively, the selection could be parallel to the search. A possible implementation of such a DSS, illustrated on Fig. 11, has two feedback loops. In the *analysis loop*, data from the problem is used to estimate the measures and predict the best algorithm. In the *optimization loop*, data from the problem is used in the algorithm portfolio to generate new candidates. The selection mechanism acts as a switch that allows a candidate to be evaluated if it corresponds to the selected algorithm.

The portfolio may initialize the algorithms in parallel, and then switch them on and off as data is collected. A first set of measures can be made during the algorithm initialization stage, which are improved during the search. However, the bias on the measures increases as the search progresses, because the candidates are generated from limited areas of the input space. This is perhaps the most difficult limitation to overcome if the system is to be implemented.

Most ELA methods are heuristics; hence, the evidence supporting their theoretical soundness is scarce. Hence, it may be unproductive to spend most of the budget on the ELA methods. Allowing a higher error on the measures requires an understanding of the error magnitude and its impact on the DSS performance, which may be significant. Such understanding could be gained by analyzing the measures as random variables dependent on the problem instance, f, and sample size, n. The error is represented by the variance, which should converge to zero when $n \to \infty$, otherwise it depends on f and n. Studies of the stochastic convergence of each measure are dependent on f; hence, they may lack practical relevance. However, more pragmatic approaches may uncover flaws on the ELA methods. For example, in [103], bootstrapping, significance tests, and meta-analysis techniques demonstrated that some measures had the same value regardless of the problem instance, sample size or dimension. Accuracy analysis is lacking for most ELA methods, even though it should be an integral part of the validation.

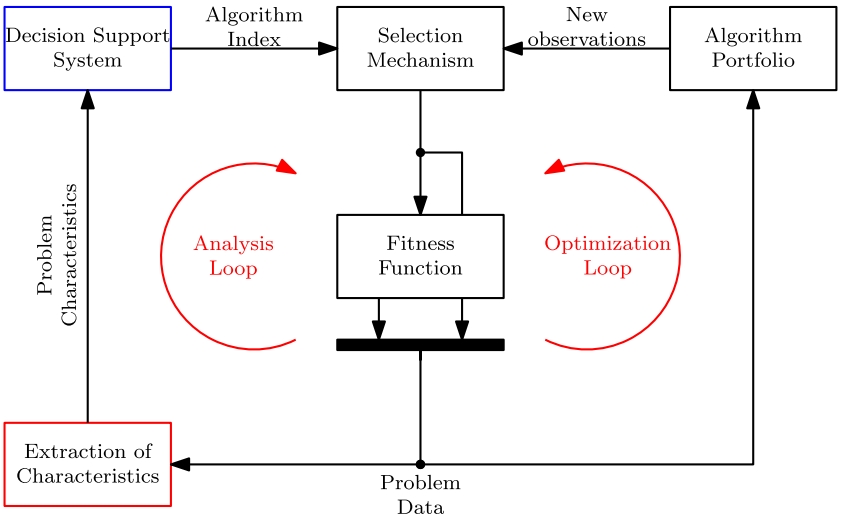


Fig. 11. Proposed structure for an on-line algorithm selection system, which has two feedback loops: In the *analysis loop*, the Decision Support System selects an algorithm from the portfolio. In the *optimization loop*, the portfolio generates new candidates.

Similarly, due to their questionable theoretical soundness, an ELA method may include assumptions that are incorrect or limiting. For example, several ELA methods use Euclidean distance to measure differences between candidates. However, this distance metric quickly converges to a constant value as the dimension increases [99]. Challenging these assumptions may lead to theoretical sound methods. Alternatively, there may be other mathematic fields which have studied the problem of characterizing a function. Inspiration for new ELA methods could be drawn from fields such as topology and differential geometry.

Although there are calls to adapt methods from discrete to continuous search spaces [86], the results may be unsatisfactory due to the lack of clearly defined neighborhoods. For example, in [101] a cheaper estimator of the size of the basins of attraction [17,19,40,123,158] was proposed. The neighborhood was defined using Delaunay triangulation, limiting the method to lower dimensions. Although a binary partition would address this limitation, a parameter is added to control the neighborhood size. Hence, the resulting measures would be dependent on this parameter, along with f and n.

The work in [1,14,102] is limited by the size of the knowledge base, which could be expanded by adding of new instances. One approach is to collect instances from libraries such as CUTEst [43]. However, this does not guarantee that the new instances are dissimilar with the ones already existing in the knowledge base. Alternatively, new instances may be evolved using a generator [140]. A visual representation of the problem space would be useful to identify empty areas. This representation should maintain the neighborhood structure existing in high dimensions. Furthermore, the generator should have sufficient flexibility to push the instances towards these areas.

Acknowledgements

This paper is a revised, updated and expanded version of [100]. Funding was provided by The University of Melbourne through MIRS/MIFRS scholarships awarded to Mario A. Muñoz, and the Australian Research Council through Grant No. DP120103678. We thank Prof. K. Smith-Miles for her valuable comments.

References

[1] T. Abell, Y. Malitsky, K. Tierney, Features for exploiting black-box optimization problem structure, in: Proceedings of the 7th International Conference on Learning and Intelligence Optimization (LION7), Lect. Notes Comput. Sci., 2013, pp. 30–36.
[2] S. Ali, K. Smith, On learning algorithm selection for classification, Appl. Soft Comput. 6 (2006) 119–138.
[3] E. Anderson, Markov chain modelling of the solution surface in local search, J. Oper. Res. Soc. 53 (6) (2002) 630–636.
[4] C. Ansótegui, M. Sellmann, K. Tierney, A gender-based genetic algorithm for the automatic configuration of algorithms, in: Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 142–157.
[5] A. Auger, B. Doerr, Theory of Randomized Search Heuristics, World Scientific, 2011.
[6] A. Auger, O. Teytaud, Continuous lunches are free plus the design of optimal optimization algorithms, Algorithmica 57 (1) (2010) 121–146.
[7] T. Bartz-Beielstein, Experimental Research in Evolutionary Computation, Natural Computing Series, Springer, Berlin, Heidelberg, 2006.
[8] T. Bartz-Beielstein, C. Lasarczyk, M. Preuß, Sequential parameter optimization, in: The 2005 IEEE Congress on Evolutionary Computation, vol. 1, 2005, pp. 773–780.
[9] J. Beck, E. Freuder, Simple rules for low-knowledge algorithm selection, in: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Lect. Notes Comput. Sci., vol. 3011, Springer, 2004, pp. 50–64.
[10] J. Beck, J. Watson, Adaptive search algorithms and fitness–distance correlation, in: Proceedings of the Fifth Metaheuristics International Conference, 2003, pp. 1–6.
[11] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (1) (2012) 281–305.

[12] H. Beyer, H. Schwefel, Evolution strategies: a comprehensive introduction, Nat. Comput. 1 (1) (2002) 3–52.
[13] M. Birattari, Z. Yuan, P. Balaprakash, T. Stützle, F-Race and iterated F-Race: an overview, in: T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuß (Eds.), Experimental Methods for the Analysis of Optimization Algorithms, Springer, Berlin, Heidelberg, 2010, pp. 311–336.
[14] B. Bischl, O. Mersmann, H. Trautmann, M. Preuß, Algorithm selection based on exploratory landscape analysis and cost-sensitive learning, in: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2012, pp. 313–320.
[15] C. Blum, J. Puchinger, G. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151.
[16] Y. Borenstein, R. Poli, Fitness distributions and GA hardness, in: Proceedings of Parallel Problem Solving from Nature (PPSN VIII), 2004, pp. 11–20.
[17] C. Brooks, E. Durfee, Using landscape theory to measure learning difficulty for adaptive agents, in: E. Alonso, D. Kudenko, D. Kazakov (Eds.), Adaptive Agents and Multi-Agent Systems, Lect. Notes Comput. Sci., vol. 2636, Springer, 2003. 561-561.
[18] E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, R. Qu, Hyper-heuristics: a survey of the state of the art, J. Oper. Res. Soc. 64 (12) (2013) 1695–1724.
[19] P. Caamaño, F. Bellas, J. Becerra, R. Duro, Evolutionary algorithm characterization in real parameter optimization problems, Appl. Soft Comput. (0) (2013).
[20] P. Caamaño, A. Prieto, J. Becerra, F. Bellas, R. Duro, Real-valued multimodal fitness landscape characterization for evolution, in: Neural Information Processing. Theory and Algorithms, Lect. Notes Comput. Sci., vol. 6443, Springer, 2010, pp. 567–574.
[21] M. Cavazzuti, Deterministic Optimization, Springer, Berlin, Heidelberg, 2013 (Chapter 4).
[22] M. Cavazzuti, Stochastic Optimization, Springer, Berlin, Heidelberg, 2013 (Chapter 5).
[23] X. Chen, Y. Ong, M. Lim, K. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (5) (2011) 591–607.
[24] A. Conn, N. Gould, P. Toint, Trust Region Methods, Society for Industrial and Applied Mathematics, 2000.
[25] A.D. Corte, K. Sörensen, Optimisation of gravity-fed water distribution network design: a critical review, Eur. J. Oper. Res. 228 (1) (2013) 1–10.
[26] C. Cortes, M. Mohri, M. Riley, A. Rostamizadeh, Sample selection bias correction theory, in: Algorithmic Learning Theory, Lecture Notes in Computer Science, vol. 5254, Springer, Berlin, Heidelberg, 2008, pp. 38–53.
[27] S. Das, P. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.
[28] Y. Davidor, Epistasis variance: a viewpoint on ga-hardness, in: G. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kauffmannn, 1991, pp. 23–35.
[29] K. De Jong, Parameter setting in EAs: a 30 year perspective, in: F. Lobo, C. Lima, Z. Michalewicz (Eds.), Parameter Setting in Evolutionary Algorithms, Stud. Comput. Intell., vol. 54, Springer, 2005, pp. 1–18.
[30] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: an autocatalytic optimizing process, Tech. Rep. 91-016, Politecnico di Milano, 1991.
[31] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[32] B. Efron, R. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, Inc., 1993.
[33] A. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evol. Comput. 3 (2) (1999) 124–141.
[34] A. Eiben, Z. Michalewicz, M. Schoenauer, J. Smith, Parameter control in evolutionary algorithms, in: F. Lobo, C. Lima, Z. Michalewicz (Eds.), Parameter Setting in Evolutionary Algorithms, Stud. Comput. Intell., vol. 54, Springer, 2005, pp. 19–46.
[35] A. Eiben, S. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms, Swarm Evol. Comput. 1 (1) (2011) 19–31.
[36] A. Eremeev, C. Reeves, On confidence intervals for the number of local optima, in: Applications of Evolutionary Computing, Lect. Notes Comput. Sci., vol. 2611, Springer, 2003. 115-115.
[37] C. Fonlupt, D. Robilliard, P. Preux, A bit-wise epistasis measure for binary search spaces, in: Proceedings of Parallel Problem Solving from Nature (PPSN V), Lect. Notes Comput. Sci., vol. 1498, 1998, pp. 47–56.
[38] O. Francois, C. Lavergne, Design of evolutionary algorithms – a statistical perspective, IEEE Trans. Evol. Comput. 5 (2) (2001) 129–148.
[39] M. Gagliolo, J. Schmidhuber, Learning dynamic algorithm portfolios, Ann. Math. Artif. Intel. 47 (2006) 295–328.
[40] J. Garnier, L. Kallel, Efficiency of local search with multiple local optima, SIAM J. Discrete Math. 15 (1) (2002) 122–141.
[41] C. Gomes, B. Selman, Algorithm portfolios, Artif. Intell. 126 (1–2) (2001) 43–62.
[42] C. Gomes, B. Selman, N. Crato, H. Kautz, Heavy-tailed phenomena in satisfiability and constraint satisfaction problems, J. Autom. Reason. 24 (1–2) (2000) 67–100.
[43] N. Gould, D. Orban, P. Toint, CUTEst: a constrained and unconstrained testing environment with safe threads, Tech. Rep. RAL-TR-2013-005, Science and Technology Facilities Council, 2013.
[44] M. Graff, H. Escalante, J. Cerda-Jacobo, A. Gonzalez, Models of performance of time series forecasters, Neurocomputing 122 (0) (2013) 375–385. advances in cognitive and ubiquitous computing Selected papers from the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012).
[45] M. Graff, R. Poli, Practical performance models of algorithms in evolutionary program induction and other domains, Artif. Intell. 174 (2010) 1254–1276.
[46] M. Graff, R. Poli, J. Flores, Models of performance of evolutionary program induction algorithms based on indicators of problem difficulty, Evol. Comput. 21 (4) (2013) 533–560.
[47] J. Grobler, A. Engelbrecht, G. Kendall, V. Yadavalli, Alternative hyper-heuristic strategies for multi-method global optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC2010), 2010, pp. 1–8.
[48] N. Hansen, A. Auger, S. Finck, R. Ros, Real-parameter black-box optimization benchmarking BBOB-2010: Experimental setup, Tech. Rep. RR-7215, INRIA, September 2010.
[49] N. Hansen, A. Auger, R. Ros, S. Finck, P. Pošík, Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009, in: Genetic and Evolutionary Computation Conference, 2011, pp. 1689–1696.
[50] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, A. Auger, Impacts of invariance in search: when CMA-ES and PSO face ill-conditioned and non-separable problems, Appl. Soft Comput. 11 (8) (2011) 5755–5769.
[51] E. Hart, J. Timmis, Application areas of ais: the past, the present and the future, Appl. Soft Comput. 8 (1) (2008) 191–201.
[52] M. Hauschild, M. Pelikan, An introduction and survey of estimation of distribution algorithms, Swarm Evol. Comput. 1 (3) (2011) 111–128.
[53] J. He, C. Reeves, C. Witt, X. Yao, A note on problem difficulty measures in black-box optimization: classification, realizations and predictability, Evol. Comput. 15 (4) (2007) 435–443.
[54] R. Heckendorn, D. Whitley, Predicting epistasis from mathematical models, Evol. Comput. 7 (1) (1999) 69–101.
[55] D. Henderson, S. Jacobson, A. Johnson, The theory and practice of simulated annealing, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 57, Springer, US, 2003, pp. 287–319.
[56] M. Hilario, A. Kalousis, P. Nguyen, A. Woznica, A data mining ontology for algorithm selection and meta-mining, in: Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery, 2009, pp. 76–87.
[57] H. Hoos, T. Stützle, Introduction, The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufman, San Francisco, 2005 (Chapter 1).
[58] H. Hoos, T. Stützle, SLS Methods, The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann, San Francisco, 2005 (Chapter 2).
[59] P. Hough, P. Williams, Modern machine learning for automatic optimization algorithm selection, in: Proceedings of the INFORMS Artificial Intelligence and Data Mining Workshop, 2006, pp. 1–6.
[60] M. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, A. Zimek, Can shared-neighbor distances defeat the curse of dimensionality?, in: M. Gertz, B. Ludäscher (Eds.), Scientific and Statistical Database Management, Lect. Notes Comput. Sci., vol. 6187, Springer, 2010, pp. 482–500.
[61] A. Howe, E. Dahlman, C. Hansen, M. Scheetz, A. Mayrhauser, Exploiting competitive planner performance, in: S. Biundo, M. Fox (Eds.), Recent Advances in AI Planning, Lecture Notes in Computer Science, vol. 1809, Springer, Berlin, Heidelberg, 2000, pp. 62–72.

[62] F. Hutter, Y. Hamadi, H. Hoos, K. Leyton-Brown, Performance prediction and automated tuning of randomized and parametric algorithms, in: Proceedings of Principles and Practice of Constraint Programming – CP 2006, Lect. Notes Comput. Sci., vol. 4204, Springer, 2006, pp. 213–228.
[63] F. Hutter, H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: C. Coello (Ed.), Learning and Intelligent Optimization, Lecture Notes in Computer Science, vol. 6683, Springer, Berlin, Heidelberg, 2011, pp. 507–523.
[64] F. Hutter, H. Hoos, K. Leyton-Brown, T. Stützle, ParamILS: an automatic algorithm configuration framework, J. Artif. Intell. Res. 36 (2009) 267–306.
[65] F. Hutter, L. Xu, H. Hoos, K. Leyton-Brown, Algorithm runtime prediction: methods & evaluation, Artif. Intell. 206 (0) (2014) 79–111.
[66] T. Jansen, On classifications of fitness functions, Tech. Rep. CI-76/99, University of Dortmund, November 1999.
[67] T. Jansen, Analyzing Evolutionary Algorithms, Springer, Berlin, Heidelberg, 2013.
[68] T. Jones, S. Forrest, Fitness distance correlation as a measure of problem difficulty for genetic algorithms, in: Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., 1995, pp. 184–192.
[69] S. Kadioglu, Y. Malitsky, M. Sellmann, K. Tierney, Isac – instance-specific algorithm configuration, in: Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, IOS Press, Amsterdam, The Netherlands, 2010, pp. 751–756.
[70] J. Kanda, A. Carvalho, E. Hruschka, C. Soares, Selection of algorithms to solve traveling salesman problems using meta-learning, Int. J. Hybrid Intell. Syst. 8 (3) (2011) 117–128.
[71] D. Karaboga, B. Akay, A survey: algorithms simulating bee swarm intelligence, Artif. Intell. Rev. 31 (1–4) (2009) 61–85.
[72] D. Karnopp, Random search techniques for optimization problems, Automatica 1 (1963) 111–121.
[73] L. Kotthoff, Algorithm selection for combinatorial search problems: a survey, AI Magazine 35 (3) (2014).
[74] K. Leyton-Brown, E. Nudelman, Y. Shoham, Learning the empirical hardness of optimization problems: the case of combinatorial auctions, in: P. Van Hentenryck (Ed.), Principles and Practice of Constraint Programming – CP 2002, Lect. Notes Comput. Sci., vol. 2470, Springer, 2002, pp. 91–100.
[75] K. Leyton-Brown, E. Nudelman, Y. Shoham, Empirical hardness models: methodology and a case study on combinatorial auctions, J. ACM 56 (2009) 22:1–22:52.
[76] E. Leyva, Y. Caises, A. González, R. Pérez, On the use of meta-learning for instance selection: an architecture and an experimental study, Inform. Sci. 266 (0) (2014) 16–30.
[77] E. Leyva, A. González, R. Pérez, Knowledge-based instance selection: a compromise between efficiency and versatility, Knowl.-Based Syst. 47 (0) (2013) 65–76.
[78] X. Li, A. Engelbrecht, M. Epitropakis, Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization, Tech. rep., RMIT University, 2013.
[79] X. Li, K. Tang, M. Omidvar, Z. Yang, K. Qin, H. China, Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization, Tech. rep., RMIT University, 2013.
[80] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, Scaling up fast evolutionary programming with cooperative coevolution, Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, Ieee, 2001, pp. 1101–1108.
[81] M. Lozano, D. Molina, F. Herrera, Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, Soft Comput. 15 (2011) 2085–2087.
[82] G. Lu, J. Li, Y. Yao, Fitness–probability cloud and a measure of problem hardness for evolutionary algorithms, in: Proceedings of the 11th European Conference on Evolutionary Computation in Combinatorial Optimization, Lect. Notes Comput. Sci., Springer, 2011, pp. 108–117.
[83] M. Lunacek, D. Whitley, The dispersion metric and the CMA evolution strategy, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA, 2006, pp. 477–484.
[84] R. Luus, T. Jaakola, Optimization by direct search and systematic reduction of the size of search region, AIChE J. 19 (4) (1973) 760–766.
[85] K. Malan, A. Engelbrecht, Quantifying ruggedness of continuous landscapes using entropy, in: Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC2009), 2009, pp. 1440–1447.
[86] K. Malan, A. Engelbrecht, A survey of techniques for characterising fitness landscapes and some possible ways forward, Inform. Sci. 241 (0) (2013) 148–163.
[87] Y. Malitsky, M. Sellmann, Instance-specific algorithm configuration as a method for non-model-based portfolio generation, in: N. Beldiceanu, N. Jussien, r. Pinson (Eds.), Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems, Lecture Notes in Computer Science, vol. 7298, Springer, Berlin, Heidelberg, 2012, pp. 244–259.
[88] Y. Malitsky, A. Sabharwal, H. Samulowitz, M. Sellmann, Non-model-based algorithm portfolios for SAT, in: K. Sakallah, L. Simon (Eds.), Theory and Applications of Satisfiability Testing – SAT 2011, Lecture Notes in Computer Science, vol. 6695, Springer, Berlin, Heidelberg, 2011, pp. 369–370.
[89] Y. Malitsky, A. Sabharwal, H. Samulowitz, M. Sellmann, Algorithm portfolios based on cost-sensitive hierarchical clustering, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI'13, AAAI Press, 2013, pp. 608–614.
[90] J. Marin, How landscape ruggedness influences the performance of real-coded algorithms: a comparative study, Soft Comput. 16 (4) (2012) 683–698.
[91] O. Mersmann, B. Bischl, H. Trautmann, M. Preuß, C. Weihs, G. Rudolph, Exploratory landscape analysis, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 829–836.
[92] O. Mersmann, M. Preuß, H. Trautmann, Benchmarking evolutionary algorithms: towards exploratory landscape analysis, in: Proceedings of Parallel Problem Solving from Nature (PPSN XI), Lect. Notes Comput. Sci., vol. 6238, Springer, 2010, pp. 73–82.
[93] P. Merz, Advanced fitness landscape analysis and the performance of memetic algorithms, Evol. Comput. 12 (3) (2004) 303–325.
[94] Z. Michalewicz, Quo vadis, evolutionary computation? on a growing gap between theory and practice, in: Advances in Computational Intelligence, No. 73 11 in Lect. Notes Comput. Sci., Springer, 2012, pp. 98–121.
[95] C. Müller, I. Sbalzarini, Global characterization of the CEC 2005 fitness landscapes using fitness–distance analysis, in: Applications of Evolutionary Computation, Lect. Notes Comput. Sci., vol. 6624, Springer, 2011, pp. 294–303.
[96] M. Montes de Oca, T. Stutzle, M. Birattari, M. Dorigo, Frankenstein's PSO: a composite particle swarm optimization algorithm, IEEE Trans. Evol. Comput. 13 (5) (2009) 1120–1132.
[97] A. Moraglio, Towards a geometric unification of evolutionary algorithms, Ph.D. thesis, University of Essex, 2007.
[98] R. Morgan, M. Gallagher, Length scale for characterising continuous optimization problems, in: Proceedings of Parallel Problem Solving from Nature (PPSN XII), Lect. Notes Comput. Sci., 2012, pp. 407–416.
[99] R. Morgan, M. Gallagher, Sampling techniques and distance metrics in high dimensional continuous landscape analysis: limitations and improvements, IEEE Trans. Evol. Comput. PP (99) (2013). 1-1.
[100] M. Muñoz, M. Kirley, S. Halgamuge, The algorithm selection problem on the continuous optimization domain, in: Computational Intelligence in Intelligent Data Analysis, Studies Comput. Intell., vol. 445, Springer, 2012, pp. 75–89.
[101] M. Muñoz, M. Kirley, S. Halgamuge, Landscape characterization of numerical optimization problems using biased scattered data, in: Proceedings of the 2012 Congress on Evolutionary Computation (CEC2012), 2012, pp. 1–8.
[102] M. Muñoz, M. Kirley, S. Halgamuge, A meta-learning prediction model of algorithm performance for continuous optimization problems, in: Proceedings of Parallel Problem Solving from Nature (PPSN XII), Lect. Notes Comput. Sci., vol. 7941, 2012, pp. 226–235.
[103] M. Muñoz, M. Kirley, S. Halgamuge, Exploratory landscape analysis of continuous space optimization problems using information content, IEEE Trans. Evol. Comput. 19 (1) (2015) 74–87.
[104] B. Naudts, L. Kallel, A comparison of predictive measures of problem difficulty in evolutionary algorithms, IEEE Trans. Evol. Comput. 4 (1) (2000) 1–15.
[105] B. Naudts, D. Suys, A. Verschoren, Epistasis as a basic concept in formal landscape analysis, in: T. Bäck (Ed.), Proceedings of the 7th International Conference on Genetic Algorithms, Morgan Kaufmann, 1997, pp. 65–72.
[106] F. Neumann, C. Witt, Bioinspired Computation in Combinatorial Optimization, Springer, Berlin, Heidelberg, 2010.

[107] J. Nocedal, S. Wright, Conjugate gradient methods, in: Numerical Optimization, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006, pp. 101–134 (Chapter 5).
[108] J. Nocedal, S. Wright, Fundamentals of unconstrained optimization, in: Numerical Optimization, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006, pp. 10–29 (Chapter 2).
[109] J. Nocedal, S. Wright, Quasi-newton methods, in: Numerical Optimization, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006, pp. 135–163 (Chapter 6).
[110] J. Nocedal, S. Wright, Trust-region methods, in: Numerical Optimization, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006, pp. 66–100 (Chapter 4).
[111] P. Oliveto, X. Yao, Runtime analysis of evolutionary algorithms for discrete optimization, in: Theory of Randomized Search Heuristics, World Scientific, 2011, pp. 21–52.
[112] G. Pappa, G. Ochoa, M.R. Hyde, A. Freitas, J. Woodward, J. Swan, Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms, Genet. Program. Evolvable Mach. (2013) 1–33.
[113] K. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Syst. Mag. 22 (3) (2002) 52–67.
[114] M. Pedersen, Tuning & simplifying heuristical optimization, Ph.D. thesis, University of Southampton, 2009.
[115] F. Peng, K. Tang, G. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization, IEEE Trans. Evol. Comput. 14 (5) (2010) 782–800.
[116] A.P. Piotrowski, J.J. Napiorkowski, P.M. Rowinski, How novel is the "novel" black hole optimization approach?, Inform Sci. 267 (2014) 191–200.
[117] E. Pitze, M. Affenzeller, A. Beham, S. Wagner, Comprehensive and automatic fitness landscape analysis using heuristiclab, in: EUROCAST 2011, Lect. Notes Comput. Sci., vol. 6927, 2012, pp. 424–431.
[118] E. Pitzer, M. Affenzeller, A comprehensive survey on fitness landscape analysis, in: Recent Advances in Intelligent Engineering Systems, Studies Comput. Intell., vol. 378, Springer, Berlin, Heidelberg, 2012, pp. 161–191.
[119] R. Poli, L. Vanneschi, W. Langdon, N. McPhee, Theoretical results in genetic programming: the next ten years?, Genet Program. Evolvable Mach. 11 (2010) 285–320.
[120] M. Preuß, C. Stoean, R. Stoean, Niching foundations: basin identification on fixed-property generated landscapes, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 837–844.
[121] K. Price, Differential evolution vs. the functions of the 2nd ICEO, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 1997, pp. 153–157.
[122] C. Reeves, Fitness landscapes, in: Search Methodologies, Springer, 2005, pp. 587–610.
[123] C. Reeves, A. Eremeev, Statistical analysis of local search landscapes, J. Oper. Res. Soc. 55 (7) (2004) 687–693.
[124] C. Reeves, C. Wright, An experimental design perspective on genetic algorithms, in: Foundations of Genetic Algorithms 3, Morgan Kaufmann, 1995, pp. 7–22.
[125] J. Rice, The algorithm selection problem, Advances in Computers, vol. 15, Elsevier, 1976, pp. 65–118.
[126] J. Rice, Methodology for the algorithm selection problem, in: Proceedings of the IFIP TC 2.5 Working Conference on Performance Evaluation of Numerical Software, 1979, pp. 301–307.
[127] S. Rochet, M. Slimane, G. Venturini, Epistasis for real encoding in genetic algorithms, in: Australian and New Zealand Conference on Intelligent Information Systems, 1996, pp. 268–271.
[128] S. Rochet, G. Venturini, M. Slimane, E. El Kharoubi, A critical and empirical study of epistasis measures for predicting ga performances: a summary, in: Third European Conference on Artificial Evolution, 1998, pp. 275–285.
[129] R. Ros, Real-parameter black-box optimisation: Benchmarking and designing algorithms, Ph.D. thesis, Universite Paris-Sud, December 2009.
[130] H. Rosé, W. Ebeling, T. Asselmeyer, The density of states – a measure of the difficulty of optimisation problems, in: Parallel Problem Solving from Nature – PPSN IV, Lect. Notes Comput. Sci., vol. 1141, Springer, Berlin, Heidelberg, 1996, pp. 208–217.
[131] J. Schmee, G. Hahn, A simple method for regression analysis with censored data, Technometrics 21 (4) (1979) 417–432.
[132] D. Seo, B. Moon, An information-theoretic analysis on the interactions of variables in combinatorial optimization problems, Evol. Comput. 15 (2) (2007) 169–198.
[133] D. Simon, R. Rarick, M. Ergezer, D. Du, Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms, Inform. Sci. 181 (7) (2011) 1224–1248.
[134] J. Smith, T. Fogarty, Operator and parameter adaptation in genetic algorithms, Soft Comput. 1 (2) (1997) 81–87.
[135] T. Smith, P. Husbands, P. Layzell, M. O'Shea, Fitness landscapes and evolvability, Evol. Comput. 10 (1) (2002) 1–34.
[136] K. Smith-Miles, Towards insightful algorithm selection for optimisation using meta-learning concepts, in: IEEE International Joint Conference on Neural Networks, 2008 (IJCNN 2008), (IEEE World Congress on Computational Intelligence), 2008, pp. 4118–4124.
[137] K. Smith-Miles, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Comput. Surv. 41 (1) (2009) 6:1–6:25.
[138] K. Smith-Miles, D. Baatar, B. Wreford, R. Lewis, Towards objective measures of algorithm performance across instance space, Comput. Oper. Res. 45 (0) (2014) 12–24.
[139] K. Smith-Miles, R. James, J. Giffin, Y. Tu, A knowledge discovery approach to understanding relationships between scheduling problem structure and heuristic performance, in: Learning and Intelligent Optimization, Lect. Notes Comput. Sci., vol. 5851, Springer, 2009, pp. 89–103.
[140] K. Smith-Miles, J. van Hemert, Discovering the suitability of optimisation algorithms by learning from evolved instances, Ann. Math. Artif. Intel. 61 (2011) 87–104.
[141] K. Smith-Miles, B. Wreford, L. Lopes, N. Insani, Predicting metaheuristic performance on graph coloring problems using data mining, in: E.-G. Talbi (Ed.), Hybrid Metaheuristics, Studies Comput. Intell., vol. 434, Springer, Berlin, Heidelberg, 2013, pp. 417–432.
[142] K. Sörensen, Metaheuristics—the metaphor exposed, Int. T. Oper. Res. 22 (1) (2015) 3–18.
[143] P. Stadler, Landscapes and their correlation functions, J. Math. Chem. 20 (1996) 1–45.
[144] K. Steer, A. Wirth, S. Halgamuge, Information theoretic classification of problems for metaheuristics, in: Proceedings of Simulated Evolution and Learning 20 08, Lect. Notes Comput. Sci., vol. 5361, Springer, 2008, pp. 319–328.
[145] C. Stephens, R. Poli, EC theory – "in theory", in: A. Menon (Ed.), Frontiers of Evolutionary Computation, Genet. Evol. Comput. Ser., vol. 11, Springer, 2004, pp. 129–155 (Chapter 7).
[146] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Tech. rep., NTU, Singapore and IIT, Kanpur, 2005. <http://www.bionik.tu-berlin.de/user/niko/Tech-Report-May-30-05.pdf>.
[147] K. Tang, F. Peng, G. Chen, X. Yao, Population-based algorithm portfolios with automated constituent algorithms selection, Inform. Sci. 279 (2014) 94–104.
[148] C. Thornton, F. Hutter, H. Hoos, K. Leyton-Brown, Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2013, pp. 847–855.
[149] M. Tomassini, L. Vanneschi, P. Collard, M. Clergue, A study of fitness distance correlation as a difficulty measure in genetic programming, Evol. Comput. 13 (2) (2005) 213–239.
[150] G. Uludag, A. Sima Uyar, Fitness landscape analysis of differential evolution algorithms, in: Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, ICSCCW 2009, 2009, pp. 1–4.
[151] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, S. Vérel, Fitness clouds and problem hardness in genetic programming, in: K. Deb (Ed.), Genetic and Evolutionary Computation – GECCO 2004, Lect. Notes Comput. Sci., vol. 3103, Springer, 2004, pp. 690–701.

[152] L. Vanneschi, M. Tomassini, P. Collard, M. Clergue, Fitness distance correlation in structural mutation genetic programming, in: C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, E. Costa (Eds.), Genetic Programming, Lect. Notes Comput. Sci., vol. 2610, Springer, 2003, pp. 455–464.
[153] V. Vassilev, T. Fogarty, J. Miller, Information characteristics and the structure of landscapes, Evol. Comput. 8 (1) (2000) 31–60.
[154] V. Vassilevska, R. Williams, S. Woo, Confronting hardness using a hybrid approach, in: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, ACM, New York, NY, USA, 2006, pp. 1–10.
[155] M. Črepinšek, S. Liu, L. Mernik, A note on teaching-learning-based optimization algorithm, Inform. Sci. 212 (0) (2012) 79–93.
[156] J. Vrugt, B. Robinson, J. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, IEEE Trans. Evol. Comput. 13 (2) (2009) 243–259.
[157] G. Wang, Q. Song, H. Sun, X. Zhang, B. Xu, Y. Zhou, A feature subset selection algorithm automatic recommendation method, J. Artif. Intell. Res. 47 (2013) 1–34.
[158] J. Watson, J. Beck, A. Howe, L. Whitley, Problem difficulty for tabu search in job-shop scheduling, Artif. Intell. 143 (2) (2003) 189–217.
[159] J. Watson, A. Howe, Focusing on the individual: Why we need new empirical methods for characterizing problem difficulty, Working Notes of ECAI 2000 Workshop on Empirical Methods in Artificial Intelligence, August 2000.
[160] E. Weinberger, Correlated and uncorrelated fitness landscapes and how to tell the difference, Biol. Cybern. 63 (1990) 325–336.
[161] E. Weinberger, P. Stadler, Why some fitness landscapes are fractal, J. Theor. Biol. 163 (2) (1993) 255–275.
[162] T. Weise, M. Zapf, R. Chiong, A. Nebro, Why is optimization difficult?, in: R. Chiong (Ed.), Nature-Inspired Algorithms for Optimisation, Stud. Comput. Intell., vol. 193, Springer, 2009, pp. 1–50.
[163] D. Weyland, A rigorous analysis of the harmony search algorithm: how the research community can be misled by a "novel" methodology, Int. J. Appl. Metaheur. Comput. 1 (2) (2010) 50–60.
[164] D. Whitley, A genetic algorithm tutorial, Stat. Comput. 4 (2) (1994) 65–85.
[165] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[166] L. Xu, F. Hutter, H. Hoos, K. Leyton-Brown, SATzilla: portfolio-based algorithm selection for SAT, J. Artif. Intell. Res. 32 (1) (2008) 565–606.
[167] L. Xu, F. Hutter, H. Hoos, K. Leyton-brown, Satzilla2009: an automatic algorithm portfolio for sat. solver description, in: 2009 SAT Competition, 2009.
[168] L. Xu, F. Hutter, H. Hoos, K. Leyton-Brown, Hydra-MIP: automated algorithm configuration and selection for mixed integer programming, in: Proceedings of the 18th RCRA Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion, 2011.
[169] L. Xu, F. Hutter, H. Hoos, K. Leyton-Brown, Evaluating component solver contributions to portfolio-based algorithm selectors, in: A. Cimatti, R. Sebastiani (Eds.), Theory and Applications of Satisfiability Testing, Lecture Notes in Computer Science, vol. 7317, Springer, Berlin, Heidelberg, 2012, pp. 228–241.
[170] B. Zadrozny, Learning and evaluating classifiers under sample selection bias, in: Proceedings of the Twenty-first International Conference on Machine Learning, ACM, New York, NY, USA, 2004, p. 114.
[171] M. Zakynthinaki, J. Stirling, C.C. Martínez, A.L.D. de Durana, M.S. Quintana, G.R. Romo, J.S. Molinuevo, Modeling the basin of attraction as a two-dimensional manifold from experimental data: applications to balance in humans, Chaos 20 (1) (2010) 013119.