Models to classify the difficulty of genetic algorithms to solve continuous optimization problems

Noel Rodríguez-Maya, Juan Flores, Sébastien Verel, Mario Graff

Noname manuscript No.
(will be inserted by the editor)

Models to Classify the Difficulty of Genetic Algorithms to Solve Continuous Optimization Problems

Noel E. Rodríguez-Maya · Juan J. Flores ·
Sébastien Verel · Mario Graff

Received: date / Accepted: date

Abstract What constitutes a hard optimization problem to an Evolutionary Algorithm (EA)? To answer the question, the study of Fitness Landscape (FL) has emerged as one of the most successful techniques. FL measures the landscape depicted by the problem's cost function. Fitness Landscape Analysis (FLA) uses a set of metrics to try to determine the hardness of problems; FL metrics can be divided in descriptive and dynamic. Descriptive metrics measure the intrinsic problem features, examples of these measures are ruggedness, neutrality, basins of attraction, and epistasis. Dynamic metrics measure the evolvability of EA, examples of these measures are fitness distance correlation, and negative slope coefficient. This contribution presents a procedure called *Performance Classification Models* (PCM) which creates learnings models to predict the performance exhibited by Genetic Algorithms (GA) in the solution of optimization problems in the continuous domain. PCM classifies the performance in two classes (easy or difficult). The dataset has as predictor variables,

Noel E. Rodriguez-Maya
Tecnológico Nacional de México / Instituto Tecnológico de Zitácuarp
Av. Tecnológico 186, Col. Manzanillos, Zitácuaro, Michoacán, México
E-mail: noel.rm@zitacuaro.tecnm.mx ✉

Juan J. Flores
Universidad Michoacana de San Nicolás de Hidalgo
Gral. Francisco J. Múgica S/N, Morelia, Michoacán, México
E-mail: juanf@umich.mx

Sébastien Verel
Laboratoire d'Informatique Signal et Image Universite du
Littoral Côte d'Opale, Calais, France
E-mail: verel@lisic.univ-littoral.fr

Mario Graff
INFOTEC - Centro de Investigación e Innovación en Tecnologías de la
Información y Comunicación, Cátedras CONACyT, Aguascalientes, México,
CONACyT Consejo Nacional de Ciencia y Tecnología, Dirección de Cátedras,
Insurgentes Sur 1582, Crédito Constructor, Ciudad de México 03940, México
E-mail: mario.graff@infotec.mx

FL features, and as target variable the performance exhibited by the GA. The problems used in experiments are benchmark optimization functions. A product of this approach, is a procedure to *Recommend Population Size* (RPS): given an optimization problem, RPS recommends the minimal population size to get an efficient level of performance. This work can be easily extended to use other metrics, or a different set of problems, or the use of other EA. Developing performance models for other EA, we can solve an instance of the algorithm selection problem.

Keywords Optimization · Performance · Genetic Algorithms · Fitness Landscape Analysis

1 Introduction

When researchers and practitioners of Evolutionary Algorithms (EA) face new optimization problems, their main difficulties are the determination of the best algorithm and its optimal input parameters that guarantee a good approximation to the optimal solution. The prediction of hardness of optimization problems (in EA) can help us set our expectations to get a solution for those problem, as well as to determine suitable parameter settings.

During the last decades, many researchers have tried to determine which are the features of optimization problems that make them difficult to solve. One of the first tools to analyze the features of problems and algorithms is the Fitness Landscape (FL) metaphor. FL was introduced in the biology field in the 30's by Sewall Wright (S. Wright, 1932), for the study of biology evolution (natural selection). He studied evolution through the relationship between the genotype space (organism) and its reproductive success (fitness). This can be viewed as an optimization problem: searching through search space to find the best fitness. Each position in the search space, has its corresponding fitness value, those values depict peaks, rugged and smooth areas, etc.

In optimization, FL is defined as the geometric form depicted by the cost functions, in other wods, the materialization of the genotype space (for each point in the search space, it corresponds a fitness value). Literature reports many FL techniques (in this work we refer to them as FL metrics) for the characterization (related with hardness) of optimization problems. Some of the most popular are: *Neutrality* measures the rate of neutral regions, *Ruggedness* measures multimodality by the number of peaks and valleys found in a random walk, *Basins of Attraction* obtains the rate of basins of attraction present in a FL with respect to the sample size, *Epistasis* establishes a rate of gene interactivity, *Fitness Distance Correlation* measures the deceptiveness of optimization problems, and *Negative Slope Coefficient* measures the hardness of optimization problems through their evolvability.

The aforementioned FL metrics have shown success in estimating the hardness of optimization problems, however, in isolation, their performances are weaknesses in the sense that only some optimization problems can be correctly characterized. It is necessary to establish a strongest metric or set of metrics,

that can more accurately predict the hardness of optimization problems. The metric or set of metrics to be used, must characterize different benchmark optimization problems, and not only a family of problems. This contribution presents an approach to predict the difficulty of solving optimization problems in the continuous domain using GA. This approach is called *Performance Classification Models (PCM)*. A byproduct of PCM is a recommender-system, a procedure to *Recommend Population Size* (RPS), which given an optimization problem, suggests the smallest efficient population size to be used by GA in the solution of that problem. PCM complements previous work by using a set of learning models to classify the difficulty of GA to solve continuous optimization problems in two dimensions. PCM uses as predictor variables the following FL metrics: neutrality, ruggedness, basins of attraction, epistasis, fitness distance correlation, and negative slope coefficient, and, as target variable, the performance obtained from GA experiments.

To establish the hardness of optimization problems, we approximate its difficulty with the performance of GA to solve them. We categorize the performance in two classes: easy and difficult. The experiments were performed using the Real-Coded Genetic Algorithms (RCGA). The learning models generated by PCM, are based on Random Forests. The models map from a set of problems to a set of difficulty indicators:

$$M : F \rightarrow \{\text{easy, difficult}\}$$

where F is the set of optimization problems in the continuous domain in two dimensions, and easy and difficult are the problem difficulty indicators. Based on PCM, a direct application of this approach is the procedure to Recommend Population Size (RPS): for a given optimization problem, RPS tries to establish the minimal population size that ensures the best GA performance. Results suggest a high correlation between the recommended population size and the performance exhibited by optimization problems.

This work is organised as follows: Section 2 reviews the related work, Section 3 defines Fitness Landscape Analysis and its related metrics, Section 4 presents a brief introduction to Real-coded GA, Section 5 presents the main proposal of this contribution the Performance Classification Models, Section 6 presents the main results and the Procedure to Recommend Population Size, and Section 7 presents a discussion and conclusions.

2 Related Work

Evolutionary Algorithms (EA) are bio-inspired optimization tools, capable of solving a variety of optimization problems. According to the nature of phenomena, some EA are more successful at solving certain problems. For example, in the case of combinatorial optimization problems, the well known No-Free-Lunch (NFL) theorem, says "No single optimization algorithm is at all times superior to any other" (Wolpert & Macready, 1995, 1997). It would be helpful to know a priori what algorithm from a set of them, is the best one

to solve a given optimization problem (K. Malan & Engelbrecht, 2009). The hardness of optimization problems, is related to features of both, problems, and EA's heuristic. Some of those features are the structure of problems, the internal operation of the EA's heuristic, and the sampling method, among others (Jones, 1995; Vanneschi, 2004; Vanneschi, Valsecchi, & Poli, 2009; He, Reeves, Witt, & Yao, 2007). To select the best EA for a given problem, many authors suggest the characterization of the problem through the use of *Fitness Landscape* features. Fitness Landscape (FL) was proposed by Wright in the 1930's (S. Wright, 1932), it refers to the geometric form of problems. FL uses the search space's features, and its orthogonal projection materialized by the cost function; with the use of fitness function (cost function), FL measures the number of peaks, the rate of neutral areas, the rate of basins of attraction, etc. (Volke, Bin, Zeckzer, Middendorf, & Scheuermann, 2014). E.g., for a smooth landscape with a single optimum will be relatively easy to solve for many algorithms, while a very rugged landscape, with many local optima, may be more difficult to solve (Horn & Goldberg, 1995; Kauffman & Johnsen, 1991).

Literature reports different approaches to predict the hardness of optimization problems when solved by EA. Recent works use different types of Fitness Landscape features: some authors make use of an isolated FL metric, while others make use of a set of FL metrics. Other approaches capture the dynamics of EA on fitness landscapes, looking for a relation between the dynamics and the problem difficulty. Results in approaches, vary according to problem's features and the efficacy of the FL metrics.

The following are examples of works that use a single FL metric. Grefenstette (Jones, 1995) demonstrates that deception alone is not necessary nor sufficient to ensure that a problem is difficult for GA. Naudts and Kallel (Naudts & Kallel, 2000b) use as difficulty metrics, epistasis variance and fitness distance correlation on easy and hard problems. They conclude that isolated metrics are no capable to establish the main characteristics of easy and hard problems. Reeves and Wright (Reeves & Wright, 1995) use epistasis as problem difficulty in bit-string problems using GA, their conclusions show a low relation between problem difficulty and epistasis.

Some authors, capture the search dynamics of EA on optimization problems. Vanneschi et al. (Vanneschi, Tomassini, Collard, & Vérel, 2006; Vanneschi, Tomassini, Pirola, Verel, & Mauri, 2006) proposed the use of Negative Slope Coefficient (NSC) to try to predict the hardness of Genetic Programming problems. NSC uses the term of fitness clouds, which is a 2-dimensional representation of evolvability, where the fitness of individuals are plotted against the fitness of its neighbours. The results show an accurate metric in the prediction of hardness of some family of problems in Genetic Programming. Fitness Distance Correlation (FDC) measures the level of deceptiveness of GA problems in a bit-string encoding (Jones & Forrest, 1995). According to the metric, problems below a threshold (-0.15) are considered as deceptives, the more deceptive, the harder the problem. FDC has been tested in different optimization problems without final results.

To improve the accuracy of FL metrics, some authors suggest the use of a set of metrics (mixing features of problems and algorithms). In the field of Genetic Programming (GP), Graff and Poli proposed a model of performance (Mario Graff and Riccardo Poli, 2010), estimating the performance of a GP system, for a given problem, using a set of points from the search space. Their results showed how the algorithm's features are related to its performance. Caamaño et al. (Caamaño, Prieto, Becerra, Bellas, & Duro, 2010; Caamaño, Bellas, Becerra, & Duro, 2013) proposed the use of FL features based on modality and separability to characterize benchmark optimization problems. They conclude that there exist a relation between those features and the performance exhibited by EA. Malan et al. (K. Malan & Engelbrecht, 2009; K. M. Malan & Engelbrecht, 2013) proposed the characterization of optimization problems (its static features) before trying to predict their performance using some EA; based on the problem's characteristics the users can select the best EA that solves the problem. Trujillo et al. (Trujillo, Martínez, Galván López, & Legrand, 2012; Trujillo, Martínez, López, & Legrand, 2011) use a set of FL features (dynamic and static metrics) to predict the hardness of GP problems. They use a black-box process: the FL features of a set of GP problems are passed to the ML process to generate a prediction model. Their results show an accurate model to predict the performance of GP.

Recently many works have emerged, due to the popularity of GECCO's workshop on Black Box Optimization Benchmark (BBOB) (Auger et al., 2012), where the contestants test the performance of their heuristics through the platform COmparing Continuous Optimisers (COCO) (Hansen, Auger, Finck, & Ros, 2010). COCO is a real-parameter optimization benchmark, where the users can perform systematic tests on a set of real-optimization problems. Mario Muñoz et al. (Muñoz, Kirley, & Halgamuge, 2015) proposed a classification model based on four FL features called "Information Content of Fitness Sequences (ICoFiS)"; their results showed an accurate model using a low rate of function evaluations. Olaf Mersman et al. (Mersmann et al., 2011) proposed the use of a reduced number of fitness landscape metrics (low-level features') to try to classify the BBOB real optimization problems into 6 classes, their results classified efficiently the problems. Another approach based on the cell mapping techniques (global behavior of non-linear dynamical systems) developed by Kerschke et al. (Kerschke et al., 2014). Their results show an interesting method for the construction of high level features, using a reduced number of function evaluations.

The methodology proposed in this article differs from our previous work (Rodriguez-Maya, Graff, & Flores, 2014) and complements the related work in several ways. First, in our previous work we classify the optimization problems using a mixture of FL measures and GA parameters, this is not practical in real life applications since GA attributes are not part of optimization problems. In this work we use only FL features which were computed from the optimization problems. Second, the majority of works related to the prediction of hardness of optimization problems, take into consideration problems's features (static features) or algorithms's features (dynamic features), but not

both at the same time. Third, based on the proposed models, a population size recommender was developed, which recommends the optimal population size for a given optimization problem. Fourth, this work uses a relatively large set of benchmark optimization problems 110 problems in the continuous domain in two dimensions.

3 Fitness Landscape Analysis

Currently, there is not a successful technique or set of techniques in the EA field, that ensure an accurate estimation of difficulty for the majority of optimization problems (K. Malan & Engelbrecht, 2009). One of the most promising tools to try to estimate the difficult of optimization problems, is the concept of Fitness Landscape (FL); generally the use of a set of FL metrics is called Fitness Landscape Analysis (FLA). FLA measures different features of optimization problems and optimization algorithms, those features that can provide some insight about the hardness of optimization problems when are solved by EA (K. Malan & Engelbrecht, 2009).

Ruggedness, smoothness, basins of attraction, and deceptiveness appear as some of the most important features to relate the difficulty of optimization problems. However, in isolation, these features are not sufficient to describe the difficulty of optimization problems when solved by EA (Caamaño et al., 2010; Jones, 1995). It is necessary to use a set of FL features, that ensure a more accurate predictions. This approach uses different difficulty metrics to measure different features of optimization problems. The set must capture the most representative features of problems and the evolutiveness of EA. To do this, we propose the use of two types of FL metrics: descriptive and dynamic (K. Malan & Engelbrecht, 2009; Merkuryeva G., 2011; Reidys & Stadler, 2002). Descriptive metrics focus on the problem's features while dynamic metrics focus mainly on the algorithm.

3.1 Terminology

The following glossary states the general terms used in the following concepts and definitions included in the remainder of the paper.

Ω - search space on the $\mathbb{R}^{\mathcal{D}}$ domain.
$\mathcal{D}$ - problem dimension.
$\mathcal{S} = \{s_1, \ldots, s_m\}$ - set of points sampled from the search space ($\mathcal{S} \subset \Omega$).
$f : \Omega \to \mathbb{R}$ - fitness function
ϵ - precision required in GA.
n - population size used in GA.
n_T - number of independent executions of GA.
$\mathcal{N}_{\mathcal{S}}(s, \delta) = \{s' \in \mathcal{S} | s \neq s' \wedge d_E(s, s') \leq \delta\}$ - the neighborhood function defined on $\mathcal{S}$, where δ is the maximum euclidean distance between s and its neighbours.

$d_E(x, y)$ - Euclidean Distance in Ω.
δ - radious of a neighbourhood in $\mathcal{S}$.

3.2 Descriptive metrics

Descriptive metrics measure some of the main features of optimization problems. The descriptive metrics used in this paper are: *Ruggedness*, *Neutrality*, *Basins of attraction*, and *Epistasis*. The rest of the sub-section describes those metrics.

3.2.1 Neutrality

Neutrality was introduced in biological evolution theory by Kimura (Kimura, 1983). In the field of EA, neutral regions are areas of the FL that have similar fitness values (López & Poli, 2006), i.e. similar fitness values within a neighborhood. Neutrality is the rate of neutral areas in $\mathcal{S}$, Equation 1 computes an estimate of neutrality based on a sample $\mathcal{S}$.

$$\text{neutrality}_{\mathcal{S}}(\delta, \gamma) = \frac{\sum_{s \in \mathcal{S}} \frac{|\mathcal{NN}_{\mathcal{S}}(s,\delta,\gamma)|}{|\mathcal{N}_{\mathcal{S}}(s,\delta)|}}{|\mathcal{S}|} \quad (1)$$

where $\mathcal{S}$ is a set of points from the search space, δ is the maximum distance between neighbours, and γ is the maximum distance between two fitnesses considered as similar, $\mathcal{NN}_{\mathcal{S}}(\cdot)$ is the neutral neighborhood function (defined in Equation 2), and $\mathcal{N}_{\mathcal{S}}(\cdot)$ is the neighborhood function defined in section 3.1.

$$\mathcal{NN}_{\mathcal{S}}(s, \delta, \gamma) = \{\forall s' \in \mathcal{S} | s \neq s' \wedge d_E(s, s') \leq \delta \wedge d_E(f(s), f(s')) \leq \gamma\} \quad (2)$$

High rates of neutrality, are not desirable in an FL to produce a suitable evolutive environment (Smith, Philippides, Husbands, & O'Shea, 2002), i.e. neutrality can affect the distribution of local optima and as a consequence the success of searching (K. M. Malan & Engelbrecht, 2013).

3.2.2 Ruggedness

Ruggedness is a measure related to the number of peaks surrounded by valleys in a FL; a problem has high degree of ruggedness when the fitness function in the search space has a high rate of changes (Vassilev, Miller, & Fogarty, 1999; Lobo, Miller, & Fontana, 2004; Pitzer & Affenzeller, 2012). In a rugged FL, the individuals of many EA can get trapped in local optima as a consequence of premature convergence (K. M. Malan & Engelbrecht, 2013); generally, the more rugged a function, the harder it is to optimize (Weise, 2009). There are many techniques to measure the level of ruggedness (Vassilev et al., 1999; K. Malan & Engelbrecht, 2009), this work uses entropy to estimate the rate of ruggedness (K. Malan & Engelbrecht, 2009). Entropy measures ruggedness

by means of three-point paths; a 3-point path is: neutral when the points have similar fitnesses, smooth when the fitnesses of points change in one direction, and rugged when the fitnesses of points change in two directions (K. Malan & Engelbrecht, 2009).

To compute the rate of ruggedness it is necessary to consider the sequence $\{\phi_t\}_{t=0}^{n}$ of fitness values picked from a simple random walk on Ω. The aim is to extract information from that sequence of shapes. The information is represented by a string $S(\gamma) = s_1 s_2 s_3 ... s_n$ of symbols $s_i \in \{\bar{1}, 0, 1\}$ obtained by Equation(3).

$$s_i = \Psi_{\phi_t}(i, \gamma) = \begin{cases} \bar{1}, & \text{if } \phi_i - \phi_{i-1} < -\gamma \\ 0, & \text{if } |\phi_i - \phi_{i-1}| \leq \gamma \\ 1, & \text{if } \phi_i - \phi_{i-1} > \gamma \end{cases} \tag{3}$$

The parameter γ is a real number that determines the accuracy of the calculation for $S(\gamma)$. Equation (4) estimates the rate of ruggedness through the entropic measure $H(S)$ exhibited by the sequence S.

$$H(S) = -\sum_{p \neq q} P_{[pq]} log_6 P_{[pq]} \tag{4}$$

where p and q are elements from the set $\{\bar{1}, 0, 1\}$, and the number 6 in the log function represents all possible shapes of the sequence. $H(S) \in [0, 1]$ is a rate of the variety of shapes present in the Fitness Landscape. The higher the value of $H(S)$, the wider the variety of rugged shapes in S (K. Malan & Engelbrecht, 2009). $P_{[pq]}$ is calculated according to Equation 5:

$$P_{[pq]} = \frac{n_{[pq]}}{n} \tag{5}$$

where $n_{[pq]}$ is the number of sub-blocks pq in the sequence $S(\gamma)$. For each rugged element, $P_{[pq]}$ calculates the probability of occurrence of that element.

3.2.3 Basins of Attraction

Basins of attraction are areas in the search space that lead to a local optimum (Pitzer, Affenzeller, & Beham, 2010). That is, a basin of attraction is a region containing a single locally optimal attractor, where all the points contained in it, are attracted by the basin (Xin, Chen, & Pan, 2009). In this work, we approximate the set of basins of attractions by the proportion of local optimal found in $\mathcal{S}$; Equation 6 computes the proportion of local optima found in $\mathcal{S}$ based on the neighborhood $\mathcal{N}_{\mathcal{S}}$.

$$\mathcal{B}_{\mathcal{S}}(\delta) = \frac{|\{\mathcal{H}(s, \delta); s \in \mathcal{S}\}|}{|\mathcal{S}|} \tag{6}$$

δ is the maximum distance between neighbours, $\mathcal{H}(\cdot)$ is a hill-climber algorithm that calculates the number of local optimums (attractor) for each point in $\mathcal{S}$ (see Equation 7).

$$\mathcal{H}(s,\delta) = \{s' \in \mathcal{N}_{\mathcal{S}}(s,\delta) | f'^{*} \leq f(s')\} \tag{7}$$

where $\mathcal{N}_{\mathcal{S}}(\cdot)$ is the neighbourhood function defined in Section 3.1, and f'^{*} is the minimum fitness value found in the neighborhood. A procedure for determining the attractors is mentioned in (Ochoa, Tomassini, Vérel, & Darabos, 2008).

3.2.4 Epistasis

Epistasis was introduced in GA by Davidor (Davidor, 1990) as an indication of problem difficulty. Epistasis is defined as the effect of one gene being dependent on the presence of one or more modifier genes, that is, the effects of a set of genes caused on another set of genes (the gene interaction); in a fitness function, this metric measures the level of separability of variables. It is possible to measure the level of epistasis through an Analysis of Variance (ANOVA). An analysis of variance measures the level of the contribution of factors (variables) in a model (fitness function); in this case we are interested in the interaction between factors. Chan et al. (Chan, Aydin, & Fogarty, 2003) have adapted ANOVA on optimization problems in continuous domains. To measure the variance, the variability of fitness values are measured by the sum of square deviations from the mean fitness (SS), partitioned in its orthogonal components. To measure the level of epistasis in optimization problems (in two dimensions), we are interested in getting to know the level of interaction between variables x and y. Equation 8 measures the level of epistasis (contribution) of the variables x and y in a cost function f.

$$SS_{xy} = \frac{1}{|\mathcal{S}|}\sum_{x\in\mathcal{S}}\sum_{y\in\mathcal{S}} f(x,y) - \frac{1}{|\mathcal{S}|^2}(\sum_{x\in\mathcal{S}}\sum_{y\in\mathcal{S}} f(x,y))^2 - SS_x - SS_y \tag{8}$$

where SS_x and SS_y are the level of contribution of factors x and y into the model, and SS_{xy} is the contribution for both variables x and y. Equations 9 and 10 compute the level of contribution for the variables x and y, respectively.

$$SS_x = \frac{1}{|\mathcal{S}|}\sum_{x\in\mathcal{S}}(\sum_{y\in\mathcal{S}} f(x,y))^2 - \frac{1}{|\mathcal{S}|^2}(\sum_{x\in\mathcal{S}}\sum_{y\in\mathcal{S}} f(x,y))^2 \tag{9}$$

$$SS_y = \frac{1}{|\mathcal{S}|}\sum_{y\in\mathcal{S}}(\sum_{x\in\mathcal{S}} f(x,y))^2 - \frac{1}{|\mathcal{S}|^2}(\sum_{x\in\mathcal{S}}\sum_{y\in\mathcal{S}} f(x,y))^2 \tag{10}$$

3.3 Dynamic metrics

Dynamic metrics try to capture the difficulty of optimization problems from the point of view of the algorithm. To measure the hardness of problems, this approach considers the intrinsic features of EA, e.g. genetic distance between individuals, rate of improvement in neighbors of individuals, etc. All those

features are expressed in terms of genetic operators (e.g. selection, mutation, and crossover) or evolvability (the level of improvements between individuals and their neighbors). In this work we use two of the most successful metrics: Fitness Distance Correlation, and Negative Slope Coefficient. This subsection describes these metrics.

3.3.1 Fitness Distance Correlation

Fitness Distance Correlation (FDC), developed by Jones and Forrest (Jones & Forrest, 1995), was one of the first metrics devised to predict the difficulty of EA to solve optimization problems. FDC measures the level of deceptiveness of optimization problems; generally, deceptiveness mislead the search to local optima rather than to global optima (Chen, Hu, Hirasawa, & Yu, 2008). The main advantage of FDC, is that it has been proved, as a suitable indicator of problem difficulty in GA and Genetic Programming (Vanneschi & Tomassini, 2002; Vanneschi, Tomassini, Collard, & Vérel, 2006; Vanneschi, Tomassini, Collard, & Clergue, 2005; Pitzer & Affenzeller, 2012). Its main disadvantage is that the optimal solutions must be known a priori, which is unrealistic in real life applications (Vanneschi, Clergue, Collard, Tomassini, & Vérel, 2004; Altenberg, 1997; Naudts & Kallel, 2000a; Vanneschi et al., 2005). Nonetheless, we are using this metric because we do know the global optimum for all the problems in the training and test sets.

Let f be the function to optimize (with a global optimum located at x^*), $\mathcal{S}$ a set of n individuals scattered through the function's domain, $\Phi = \{\phi_1, ..., \phi_n\}$ the corresponding evaluations of the objective function at those points, and $D = \{d(x_1, x^*), ..., d(x_n, x^*)\}$ the distance of the individuals to the global optimum. fdc is defined by Equation 11.

$$fdc = \frac{C_{\Phi D}}{\sigma_\Phi \sigma_D} \tag{11}$$

where σ_Φ and σ_D standard deviations, $\bar{\phi}$ and $\bar{d}$ means of Φ and D, respectively, and the covariance of Φ and D is defined by Equation 12.

$$C_{\Phi D} = \frac{1}{n}\sum_{i=1}^{n}(\phi_i - \bar{\phi})(d_i - \bar{d}) \tag{12}$$

3.3.2 Negative Slope Coefficient

NSC was developed by Vanneschi et al. (Vanneschi, Tomassini, Collard, & Vérel, 2006) to capture the evolvability of EA; evolvability is the capacity of genetic operators to improve the fitness quality of individuals (Pitzer & Affenzeller, 2012). To measure evolvability, NSC uses the concept of Fitness Cloud (FC): the fitnesses of individuals against the fitnesses their neighbors are plotted creating a cloud of evolvability (the level of improvement between individuals and neighbors) (Vérel, Collard, & Clergue, 2007). In a FC each set of individuals (bins) creates a cloud, that cloud represents the neighbors'

improvements; all the clouds have a centroid (the mean of fitness for the x and y axis), those centroids serve as points to trace a line and its related slope is the key of this measure. The main disadvantages of the usage of NSC is the fact that its values are not normalized (Pitzer & Affenzeller, 2012; Vanneschi, Tomassini, Collard, & Vérel, 2006), and that NSC does not converge for large samples (Vanneschi, Verel, Tomassini, & Collard, 2009).

To compute NSC, we use $\mathcal{S}$ as an approximation of the search space Ω. Let $\mathcal{S}$ be a set of individuals, f is a fitness function that assigns a real value to each individual x, and $V_{x_j} = \{v_1^j, v_2^j, ..., v_{m_j}^j\}$ the set of neighbors of a given individual $x_j, \forall j \in [1, n]$. The neighbors are obtained by applying one step of a genetic operator. The FC can be visualized as a plot where abscissas are the set of all individuals' fitnesses, and the ordinates the fitnesses of their neighbors, see Equation (13).

$$FC = \{(f(x_j), f(v_k^j)), \forall j \in [1, n], \forall k \in [1, m_j]\} \tag{13}$$

where n is the number of individuals and m is the number of predefined neighbors for each individual.

Once the fitness cloud is determined, each element of abscissas and ordinates are split into k segments $\{I_1, I_2, ..., I_k\}$, $\{J_1, J_2, ..., J_k\}$. Then, the averages of abscissae $\{M_1, M_2, ..., M_k\}$ and ordinates, $\{N_1, N_2, ..., N_k\}$ are calculated. The segment set $S = \{S_1, S_2, ..., S_{k-1}\}$, where each S_i connects the points (M_i, N_i) to point (M_{i+1}, N_{i+1}) is created. The slope set P is calculated, where $P_i = (N_{i+1} - N_i)/(M_{i+1} - M_i)$, $\forall i \in [1, k-1]$. The Negative Slope Coefficient is computed by Equation 14.

$$nsc = \sum_{i=1}^{k-1} min(0, P_i) \tag{14}$$

Vanneschi et al. (Vanneschi, Tomassini, Collard, & Vérel, 2006) proposed the following hypothesis with respect to *nsc*: negatives values correspond to difficult problems, and values equal to 0 correspond to easy problems.

4 Real-Coded Genetic Algorithms

Genetic Algorithms (GA) is a population-based optmization method, invented by John Holland in the 1960's (Holland, 1992). The GA original representation is based on bit-strings; this representation can solve a variety of problems including problems in continuous search spaces. However, for many optimization problems (e.g. real-coded problems), it is necessary a codification-decodification process, this has as consequence a high computational cost and low precision. Alternatively, its real-encoding representation, called Real-Coded Genetic Algorithms (RCGA) (A. H. Wright, 1991), can be used with problems in continuous domains (Herrera, Lozano, & Verdegay, 1998).

The general procedure for the RCGA consists of three basic operations (Herrera et al., 1998):

1. Evaluation of individual's fitness,
2. Formation of a gene pool (intermediate population) through selection,
3. Recombination through crossover and mutation operators.

The mechanisms to evolve populations are called Genetic Operators (GO). GO can be defined as the heuristic procedures that guide the optimization process within an EA. The behavior (and success) of a RCGA is also related with the type of genetic operators and with the numerical values of its input parameters (e.g. population size, crossover rate, mutation rate, etc.) (A. H. Wright, 1991).

GA is the metaheuristic used in this work. The performance of this optimization metaheuristic to solve a problem, expressed as success rate, will be used as a measure of the problem's difficulty. We assume that easy problems are those that can be solved with a high success rate. On the other hand, hard problems are those whose solution is seldom found, exhibiting a low success rate.

5 Performance Classification Models

This section presents a procedure called *Performance Classification Models* (*PCM*), PCM creates learning models to classify the performance obtained by GAs to solve continuous optimization problems in two dimensions. The procedure and models, are based on a set of optimization problems (110 problems) in the continuous domain in two dimensions, whose FL have different features. Without loss of generality, we are only considering minimization problems in this article.

The learning models are based on supervised methods, particularly *Random Forests*[1]. The models use as predictor variables, fitness landscape features derived from the objective function of optimization problems; we use the FL metrics described in the previous section: neutrality, ruggedness, basins of attraction, epistasis, fitness distance correlation and negative slope coefficient. As we mentioned in section 3, these metrics capture both, problems's and algorithm's features.

The optimization method used to derive performance and therefore difficulty is the Real-Coded Genetic Algorithm (RCGA). To assign a difficulty measure for each problem, we approximate the difficulty of the problems with the performance obtained by the GA to solve it. The performance is the success rate for reaching the global optimal (given an error margin) by the GA. Then, the performance is discretized and classified into two categories: easy, and difficult. The target values for the learning models, are the (discretized) performances obtained by the GA.

The procedure can be extended easily to use other FL metrics, problems in higher dimensions, the incorporation of other Evolutionary Algorithms, the

[1] Several learning methods were tested and Random Forests was the one with the best classification performance.

use of a finer classification granularity, and even the creation of applications (e.g. the system to recommend population sizes). The following paragraphs explain in more detail the proposed procedure.

5.1 Problem statement

In this contribution, we propose to solve the following problem: given an optimization problem involving function f, derive models capable of predicting the difficulty encountered by GA when solving it. Formally, the models M_n (created by PCM) are supervised models that map from a set of problems F to a set of difficulty indicators DI. See Equation (15).

$$M_n : F \rightarrow \{easy,\ difficult\} \tag{15}$$

M_n refers to a learning model developed using a population size n. F is a set of continuous optimization problems.

We define performance as the rate of successful trials on which the global optimum was found in GA experiments. We set the number of trials, n_T to 100 and the maximum number of generations of the GA to $1,000$. Performance, P, has two parameters (related to the GA operation): population size (n) and precision (ϵ): for different population sizes and/or different precisions, we get different performances. For each $f \in F$, we consider the sets of population sizes $\mathcal{N} = \{50i\}, i \in [1, 10]$ and precisions $\mathcal{E} = \{10^{-i}\}, i \in [5, 10]$. Perfomance is defined by Equation 16.

$$P_{n,\epsilon}(f) = \frac{|\{x | x = GA_{n,\epsilon}(f) \text{ and } |f(x) - f^*| \leq \epsilon\}|}{n_T} \tag{16}$$

where x is the solution returned by $GA_{n,\epsilon}(f)$ — the GA-based problem solver with a population size n and a required precision ϵ — and n_T is the number of trials. Given a performance value p, $DI(p)$ discretizes its value according to Equation (17). We decided to call difficult to all problems whose performance is smaller than or equal to 0.5, and easy the rest of them.

$$DI(p) = \begin{cases} \text{difficult}, & \text{if } 0 \leq p \leq \frac{1}{2} \\ \text{easy}, & \text{if } \frac{1}{2} < p \leq 1.0 \end{cases} \tag{17}$$

The function MoD (Metrics of Difficulty) (defined in Equation 18), computes the FL metrics (computation details in Section 3) for a given problem, specified by its objective function.

$$MoD : F \rightarrow \mathbb{R}^6 \tag{18}$$

where F are optimization problems in the continuous domain, and $\mathbb{R}^6$ is a 6-dimensional vector in the $\mathbb{R}$ domain.

5.2 Dataset

The dataset $\mathcal{D}$ is defined by the couple $\mathcal{D} = \langle \mathcal{M}, \mathcal{P} \rangle$, where $\mathcal{M}$ are the fitness landscape features of F (see Equation 18), and $\mathcal{P}$ are the performance values of GA when solving functions in F (see Equation 17). The idea is to construct as many models as population sizes; i.e., 10 learning models. To construct a learning model M_n which considers a specific population size n, its predictor variables $\mathcal{M}_n$ are the fitness landscape features of F, and its target values $\mathcal{P}_n$ are the *average performance*, the average performance is computed from the set of performances described by $GA_{n,\epsilon \in \mathcal{E}}$. Algorithm 1, *Dataset*, generates the training set to produce a model M_n.

Algorithm 1 Dataset(n)

1: $F = \{110 \text{ optimization problems}\}$;
2: $\mathcal{E} = \{1 \times 10^{-5}, 1 \times 10^{-6}, ..., 1 \times 10^{-10}\}$;
3: $\mathcal{M} = \langle\rangle$;
4: $\mathcal{P} = \langle\rangle$;
5: **for** $f \in F$ **do**
6: $\quad sum = 0$;
7: $\quad$ **for** $\epsilon \in \mathcal{E}$ **do**
8: $\quad\quad sum = sum + P_{n,\epsilon}(f)$;
9: $\quad$ **end for**
10: $\quad avg = \frac{sum}{|\mathcal{E}|}$;
11: $\quad \mathcal{M} = append(\mathcal{M}, MoD(f))$;
12: $\quad \mathcal{P} = append(\mathcal{P}, DI(avg))$;
13: **end for**
14: return $\langle \mathcal{M}, \mathcal{P} \rangle$;

F, and $\mathcal{E}$ are the sets of optimization problems, and level of precisions, respectively (lines 2 and 3). $\mathcal{M}$ and $\mathcal{P}$ are the initial sequences (predictor and target) to be computed (initialized in lines 3 and 4). For each $f \in F$ and for each $\epsilon \in \mathcal{E}$ the average of performance is computed (lines 5-10). For each $f \in F$, a 6-dimensional array representing its FL features is computed through the MoD function (see Equation 18) and stored in $\mathcal{M}$ (line 11), and its corresponding average performance is discretized through DI function (see Equation 17) and stored in $\mathcal{P}$ (line 12), once all the functions were considered to form the dataset, the dataset $\langle \mathcal{M}, \mathcal{P} \rangle$ is returned by the procedure (line 14).

5.3 Model

The generated models are based on Random Forest method, which is a ensemble of random decision trees. Algorithm 2 shows the basic operation of Random Forest (Breiman, 2001).

Algorithm 2 RF_Ensemble($\mathcal{D}$)

1: **for** b=1 to B **do**
2: select a bootstrap sample d of size m from $\mathcal{D}$
3: build a random decision tree T_b using d:
4: recursively repeat the following steps for each terminal node of T_b:
5: a. pick the best variable and set it as split-point.
6: b. split the node into two child nodes.
7: **end for**
8: return the ensemble of trees$\{T_b\}_1^B$

The ensemble of decision trees are composed by B decision trees (line 1), to build the decision trees is necessary sampling the training data (line 2). The samples are selected in random and the size is an input parameter. For each decision tree, is selected (from the set of variable) the best variable (according to different criterion), and it is set as the split node (line 5), and the node is split into two child nodes (line 6). The process is repeated recursively for each node (line 4). Finally the ensemble of random trees is returned (line 7). The classification task is performed as follows: let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree, then, $\hat{C}_{rf}(x) =$ majority vote$\{\hat{C}_b(x)\}_1^B$.

6 Experimental Results

This section shows the experimental results of this proposal. The first part describes the optimization problems used in the experiments, and the parameters setting for both, Fitness Landscape metrics and for the GA metaheuristic. The second part shows the accuracy and confusion matrix obtained by the learning models. Finally, we show a practical application of this approach: the *Procedure to Recommend Population Size*.

6.1 Benchmark Problems

This work is based on a set of optimization problems in the continuous domain in two dimensions. The problems consist of 110 benchmark functions with different features; these functions are to be minimized (see Appendix A.1).

6.2 Parameters Settings

Table 1 shows the main parameters settings used to compute the Fitness Landscape metrics (defined in Section 3).

Table 1 Parameters of Fitness Landscape metrics.

FL metric	**Parameter-value**
Neutrality	$\delta = d_E(L_b, U_b) \times 0.1$, $\gamma = \lvert f^{max} - f^* \rvert \times 0.001$
Ruggedness	$\gamma = \lvert f^{max} - f^* \rvert \times 0.001$
Basins of attraction	$\delta = d_E(L_b, U_b) \times 0.1$
Epistasis	no parameters
Fitness Distance Correlation	distances = number of genetic steps to reach the optimal
Negative Slope Coefficient	evolvability = one step of genetic operators

f^{max} and f^* are the maximum and minimum fitness values, respectively, L_u and U_b are the lower and upper bounds of the search space defined for function $f \in F$. All the FL metrics use a sample of 1000 points picked at random using uniform distribution within each problem's domain. The GA's performances to solve the functions, were obtained varying the population size and precision; Table 2 shows the parameters setting used by GA.

Table 2 Parameters to determine the performance GA on problems in F.

Parameter	**Value**
Population Size (n)	$\{50i\}, i \in [1, 10]$
Precision (ϵ)	$\{1 \times 10^{-i}\}, i \in [5, 10]$
Number of Generations	$1,000$
Crossover type	Arithmetical
Crossover rate	70%
Mutation type	random (uniform)
Mutation rate	30%
Selection	Tournament of size 10
Codification	Real

To get a statistically significant measure, the experiments were repeated 100 times and the mean was reported.

6.3 Learning Models

Although several learning methods were tested, we are only reporting results obtained by *Random Forests* (Breiman, 2001), since this was the most accurate method. This work uses the *WEKA* framework (using 10 trees, and descriptive and dynamic features) (Hall et al., 2009) for developing the models.

The effectiveness of models were tested using different sets of input variables in the dataset: only descriptive metrics, only dynamic metrics, and all metrics. The target values (GA's performance) for the dataset were determined, for $f \in F$, computing the average of performance obtained for the precisions $\epsilon = 1 \times 10^{-i}$, $i \in [5, 10]$, given a population size $n \in \mathcal{N}$. Figure 1 shows the average performance for each function, for the considered population sizes. The average performance is the mean of performances obtained in

a set of performances computed through a fixed population size and different precision errors, that is, the performance for a population size is the mean of performance considering different precision errors ($\epsilon = 1 \times 10^{-i}, i \in [5, 10]$). We considered the average performance as an measure of performance since the variation of population size and error margin affects the final performance, then an indicator of performance considering those variations is the average performance.

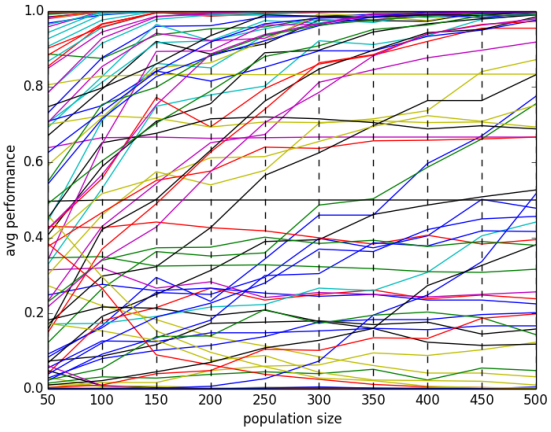

Fig. 1 Average performance for ech $f \in F$ considering different population sizes.

In Figure 1 the x axis are the population sizes, and, the y axis the normalized performance. The figure shows that the performance for some problems does not tend to improve (maintaining the performance below 0.5), while others, its performance increases at the same time that population size. This behavior in the majority of bins (dotted lines), the performances are grouped into two groups, between $0.0 \leq p \leq 0.5$ and $0.5 < p \leq 1.0$ approximately; we can consider this as an indication of problem's difficulty. This fact leads to grouping functions is two classes, according to difficulty: difficult and easy.

Table 3 shows the accuracy of the learned models; the first column (n) is the population size, considered to form the models M_n, the last three columns are the accuracy of the models obtained by the the models derived using three sets of metrics (descriptive, dynamic, and all). To train and validate the classification models, we used 10-fold cross-validation. The models with the best classification accuracy are highlighted in bold-face.

Table 3 Accuracy obtained for different population sizes (n) by models using descriptive, dynamic, and all metrics.

	Fitness Landscape Metrics		
n	**Descriptive**	**Dynamic**	**All**
50	66%	68%	**76**%
100	70%	67%	**75**%
150	68%	68%	**70**%
200	65%	66%	**74**%
250	70%	**72**%	71%
300	70%	**72**%	71%
350	70%	**73**%	70%
400	67%	66%	**73**%
450	67%	66%	**73**%
500	**70**%	65%	65%
Mean	67%	68%	**72**%

From Table 3 we can establish that the most accurate models are those based on both descriptive and dynamic FL metrics. To measure the quality of models, Tables 4 and 5 show the confusion matrices for the best ($n = 50$) and worst ($n = 500$) models, based on all FL metrics.

Table 4 Confusion matrix for the best model ($n = 50$) — input variables are all metrics and target value is performance.

class/predict	difficult	easy	**Recall**	**Precision**	**ROC Area**
difficult	47	13	0.78	0.78	0.81
easy	13	37	0.74	0.74	0.81
		Mean	**0.76**	**0.76**	**0.81**

Table 5 Confusion matrix for the best model ($n = 500$) — input variables are all metrics and target value is performance.

class/predict	difficult	easy	**Recall**	**Precision**	**ROC Area**
difficult	20	20	0.5	0.51	0.72
easy	19	51	0.73	0.72	0.72
		Mean	**0.65**	**0.64**	**0.72**

Tables 4 and 5 show the details on the precision (column 5) to classify the hardness of the problems: *easy* problems present rates between 72% and 74%, while *difficult* problems exhibit rates between 51% and 78%. These results give us some indications about the models classification capacities: in the worst case, models can predict the *difficult* problems with a precision of 51% (random capacities), and in the best case, models can predict the *easy* problems with a precision of 74%. It is important to mention that in average, the models can predict both *easy* and *difficult* problems with a precision of 72%.

6.4 Procedure to Recommend Population Size

Based on the learning models generated by PCM, a procedure to Recommend Population Size (RPS) for GA was proposed (Algorithm 3); the recommendation is based on the capacity of models to predict the hardness of optimization problems. Basically the procedure tests the models M_n (according to Equation 15) to find the output class where the optimization problem f is classified as *easy*. The population size that corresponds to that model will be the recommended population.

Algorithm 3 RPS(f)

```
1: for n ∈ (50, ..., 500) do
2:     dataset = DATASET(n) //according to Algorithm 1.
3:     M_n = RF_Ensemble(dataset) //according to Algorithm 2.
4:     if M_n(f)=easy then
5:         return n
6:     end if
7: end for
8: return undef
```

RPS has as input parameter f which is an optimization function in two dimensions, an iterative process is performed taking into consideration different population sizes (line 1), the dataset is created considering the current population size (line 2), using the dataset created in the last step, the learning model is built (line 3), the models' output is compared with the easy label, if some output matches with the easy label then, the population associated to the model is returned (line 5), otherwise, undef is returned (line 6). The returned population size can be considered as optima (in the range $50 - 500$): the problem is tested into the models $M_n, n \in [50, 500]$, and is selected the smallest n where the problem is classified as easy.

To validate the procedure to Recommend Population Size (RPS) (Algorithm 3) we compare the optimal population size against the population size recommended by RPS; we called optimal population size to such population size where the average performance (discretized in easy or difficult according to Equation 17) is easy, in other words, the smallest populations where its average performance is categorized as easy. Figure 2 shows a scatter plot between the optimal population size against the recommended population size by *RPS* for the F set.

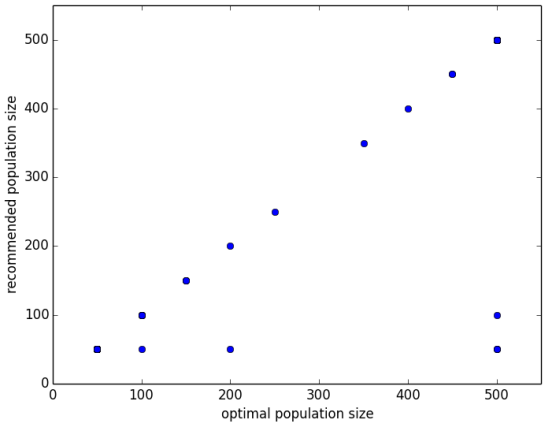

Fig. 2 Population sizes recommended by the *RPS* procedure.

Points in Figure 2 show a strong positive correlation (Pearson's Correlation, $r = 0.92$) and an accuracy of prediction of 95% between the optimal population size and the recommended population size for the F set. This indicates the accuracy exhibited by the learning models; the majority of problems in F are classified correctly.

7 Discussion and Conclusions

This section concludes the article presenting a summary of the findings presented in this contribution, followed by our conclusions.

7.1 Discussion

Generally, many approaches use isolated Fitness Landscape metrics to predict the hardness of Evolutionary Algorithms to solve optimization problems. The results were not very persuasive in the majority of cases. The performance of some metaheuristic algorithms to solve optimization problems (e.g., GA) is closely related to the geometric form of the problem's landscape. They often have problems to solve rugged landscapes. What happens when the landscape is not rugged but is deceptive?, surely some FL metrics will categorize the problem as an easy problem. However, some deceptive problems are categorized as hard problems. So it is necessary the use of a set of FL metrics that capture the majority of aspects of optimization problems: multimodality, neutrality, separability, evolvability, etc. In this approach, *PCM* uses a set of Fitness Landscape metrics to perform a Fitness Landscape Analysis; the metrics are grouped into descriptive and dynamic. While descriptive metrics measure the

statistical properties of optimization problems, the dynamic metrics capture the evolvability of the heuristic.

The models to classify the hardness of GA to solve optimization problems in two dimensions, presented in this work, are based purely on FL metrics and performances obtained by GA. It is important to remark that given a new optimization problem, the generated models can predict (easy or difficult) the performance of GA to solve it, based uniquely on the problem's FL-metrics. The generated models (using all FL metrics) obtained a mean of 72% of accuracy to classify the problems correctly. Contrary to our assumptions, in some cases, the models generated by dynamic or by descriptive metrics (M_{250}, M_{300}, M_{350}, and M_{500}) obtained slightly more accurate models than using all FL metrics. Our hypothesis about the accuracy of the models generated by the dynamic features, is about the intrinsic behavior of the metaheuristic, the larger the population size (in this case 250, 300, and 350) the greater the opportunity (perhaps, due to increased diversity in genetic material) to converge to global optimal. The number of instances considered in the dataset, play an important role to construct accurate models; the more instances, the more accurate models. In this work we used a set of 110 optimization problems, these are not enough to construct more accurate models, however, literature reports a limited number of benchmark functions.

We approximate the hardness of GA to solve optimization problems with the performance of GA to solve them; the performance is the rate of times when the global optimal was reached, given population size and precision. Literature reports other types of performances as the number of function evaluations, the expected running time, etc. We decided to use "the rate of times to get the optimal" because it is closer to the natural aim of metaheuristics (to get the global optimum) and because the result is a number between $[0, 1]$. In future works, we will develop models using other performance measures.

A second problem that can be solved using this approach is to recommend the population size that guarantees a good success rate. The proposed procedure uses the set of generated models to try to determine the smallest where the GA will be successful to solve a particular optimization problem. The first results show a strong correlation and prediction between the optimal population size and the recommended population size, having a correlation value equal to 0.92 and an accuracy of prediction of 95%. Another interesting application is to determine based only on its descriptive FL properties, which heuristic parameters (e.g. crossover rate, mutation rate, among others) could be the most appropriate to solve an optimization problem (this application will be developed in the near future). These approaches can be applied using different metaheuristics, and once these problems have been solved, we can finally propose a first approximation of the *Algorithm Selection Problem*.

7.2 Conclusions

This contribution presents a procedure called Performance Classification Models (PCM) to construct learning models to predict the difficulty of GA to solve continuous optimization problems in two dimensions. The models are based on a supervised machine learning technique: *Random Forests*. The models classify the difficulty of problems into two groups: *easy* and *difficult*. Models use as predictor variables two kinds of Fitness Landscape features: descriptive and dynamic. As target variable, it uses performance, expressed as success rate. The predictor and target variables are based on 110 continuous optimization problems in two dimensions. The models generated by PCM obtained a mean of accuracy of 72%. The generated models were used in an application called Procedure to Recommend Population Size (RPS). RPS reports a correlation (0.92) and an accuracy of prediction of 95% with respect to the optimal population size and the recommended population size.

As future directions, we are interested in the incorporation of other FL metrics and other benchmark problems. Another interesting topic is to incorporate others EA to our proposal, and other dimensions.

A Appendices

A.1 Optimization problems

The following table shows the definition of functions used in this paper, more details in (Jamil & Yang, 2013).

No.	Name	Function	Lower and Upper bounds	Global Optima
f_1	Ackley	$f(\vec{x}) = 20 + e - 20 \cdot e^{-\frac{1}{5}\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}} - e^{\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)}$	$-15 \leq x_i \leq 30, i \in [1,2]$	$f(\vec{x}^*) = 0, \vec{x}^* = (0,0)$
f_2	Beale	$f(x_1,x_2) = \left(x_1x_2^3 - x + 2.625\right)^2 + \left(x_1x_2^2 - x_1 + 2.25\right)^2 + (x_1x_2 - x_1 + 1.5)^2$	$-4.5 \leq x_i \leq 4.5, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (3, 0.5)$
f_3	Bohachevsky	$f(x_1,x_2) = x_1^2 - 0.3\cos(3\pi x_1) + 2x_2^2 - 0.4\cos(4\pi x_2)$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = -0.7$, $\vec{x}^* = (0,0)$
f_4	Booth	$f(x_1,x_2) = (2x_1 + x_2 - 5)^2 + (x_1 + 2x_2 - 7)^2$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1,3)$
f_5	Branin	$f(x_1,x_2) = \left(-0.129185x_1^2 + \frac{5x_1}{\pi} + x_2 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	$-5 \leq x_i \leq 15, i \in [1,2]$	$f(\vec{x}^*) = 0.397887$, $\vec{x}^* = (-\pi, 12.275)$
f_6	Dixon Price	$f(x_1,x_2) = 2\left(2x_2^2 - x_1\right)^2 + (x_1 - 1)^2$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1, 1.707107)$
f_7	Goldstein Price	$f(x_1,x_2) = ((3x_1^2 + 6x_1x_2 - 14x_1 + 3x_2^2 - 14x_2 + 19)(x_1 + x_2 + 1)^2 + 1)((12x_1^2 - 36x_1x_2 - 32x_1 + 27x_2^2 + 48x_2 + 18)(2x_1 - 3x_2)^2 + 30)$	$-2 \leq x_i \leq 2, i \in [1,2]$	$f(\vec{x}^*) = 3$, $\vec{x}^* = (0,-1)$
f_8	Griewank	$f(x_1,x_2) = \frac{x_1^2}{4000} + \frac{x_2^2}{4000} - \cos(x_1)\cos\left(\frac{x_2}{\sqrt{2}}\right) + 1$	$-600 \leq x_i \leq 600, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_9	Hump	$f(x_1,x_2) = \frac{x_1^6}{3} - 2.1x_1^4 + 4x_1^2 + x_1x_2 + 4x_2^4 - 4x_2^2 + 1.03163$	$-5 \leq x_i \leq 5, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0.0898, -0.7126)$
f_{10}	Michalewicz	$f(x_1,x_2) = -\sin(x_1)\sin^{20}\left(\frac{x_1^2}{\pi}\right) - \sin(x_2)\sin^{20}\left(\frac{2x_2^2}{\pi}\right)$	$0 \leq x_i \leq \pi, i \in [1,2]$	$f(\vec{x}^*) = -1.8013$, $\vec{x}^* = (2.2023, 1.57073)$
f_{11}	Rastrigin	$f(x_1,x_2) = x_1^2 + x_2^2 - 10\cos(2\pi x_1) - 10\cos(2\pi x_2) + 20$	$-5.12 \leq x_i \leq 5.12, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{12}	Rosenbrock	$f(x_1,x_2) = (x_1 - 1)^2 + 100\left(x_2 - x^2\right)^2$	$-5 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1,1)$
f_{13}	Schwefel	$f(x_1,x_2) = -x_1\sin\left(\sqrt{\lvert x_1\rvert}\right) - x_2\sin\left(\sqrt{\lvert x_2\rvert}\right) + 837.966$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1,1)$
f_{14}	Shubert	$f(x_1,x_2) = (\cos(2x_1 + 1) + 2\cos(3x_1 + 2) + 3\cos(4x_1 + 3) + 4\cos(5x_1 + 4) + 5\cos(6x_1 + 5))(\cos(2x_2 + 1) + 2\cos(3x_2 + 2) + 3\cos(4x_2 + 3) + 4\cos(5x_2 + 4) + 5\cos(6x_2 + 5))$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = -186.7309$, $\vec{x}^* = (-7.708309818, -0.800371886)$
f_{15}	Sphere	$f(\vec{x}) = \sum_{i=0}^{n-1} x_i^2$	$-5.12 \leq x_i \leq 5.12, i \in [1,2]$	$f(\vec{x}^*) = 0, \vec{x}^* = (0,0)$
f_{16}	Trid	$f(x_1,x_2) = (x_1 - 1)^2 + (x_2 - 1)^2 - x_1x_2 + 1$	$-4 \leq x_i \leq 4, i \in [1,2]$	$f(\vec{x}^*) = -1$, $\vec{x}^* = (2,2)$
f_{17}	Zakharov	$f(x_1,x_2) = (0.5x_1 + 1.x_2)^4 + (0.5x_1 + 1.x_2)^2 + x_1^2 + x_2^2$	$-5 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{18}	Dropwave	$f(x_1,x_2)=\frac{-\cos\left(12\sqrt{x_1^2+x_2^2}\right)-1}{0.5(x_1^2+x_2^2)+2}$	$-5.12\leq x_i\leq 5.12, i\in[1,2]$	$f(\vec{x}^*)=-1$, $\vec{x}^*=(0,0)$
f_{19}	Egg Holder	$f(x_1,x_2)=(-x_2-47)\sin(\sqrt{\lvert\frac{x_1}{2}+x_2+47\rvert})-x\sin(\sqrt{\lvert x_1-x_2-47\rvert})$	$-512\leq x_i\leq 512, i\in[1,2]$	$f(\vec{x}^*)=-959.6407$, $\vec{x}^*=(512,404.2319)$
f_{20}	Holder	$f(x_1,x_2)=-\lvert e^{\lvert 1-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\rvert}\cos(x_2)\sin(x_1)\rvert$	$-10\leq x_i\leq 10, i\in[1,2]$	$f(\vec{x}^*)=-19.2085$, $\vec{x}^*=(8.05502,9.66459)$
f_{21}	Levy13	$f(x_1,x_2)=(x_1-1)^2(1-\sin^2(3\pi x_2))+\sin^2(3\pi x_1)+(x_2-1)^2(\sin^2(2\pi x_2)+1)$	$-10\leq x_i\leq 10, i\in[1,2]$	$f(\vec{x}^*)=0$, $\vec{x}^*=(1,1)$
f_{22}	Styblinski Tang	$f(x_1,x_2)=0.5(x_1^4-16x_1^2+5x_1+x_2^4-16x_2^2+5x_2)$	$-5\leq x_i\leq 5, i\in[1,2]$	$f(\vec{x}^*)=-39.16599*2$, $\vec{x}^*=(-2.903534,-2.903534)$
f_{23}	Randompeaks	$f(x_1,x_2)=-2e^{-0.5((x_1-21)^2+(x_2-25)^2)}-2e^{-0.5((x_1-8)^2+(x_2-25)^2)}+5e^{-0.1((x_1-15)^2+(x_2-20)^2)}+2e^{-0.5((x_1-25)^2+(x_2-16)^2)}-2e^{-0.08((x_1-20)^2+(x_2-15)^2)}+2e^{-0.5((x_1-5)^2+(x_2-14)^2)}+3e^{-0.08((x_1-25)^2+(x_2-10)^2)}+2e^{-0.1((x_1-10)^2+(x_2-10)^2)}-2e^{-0.5((x_1-5)^2+(x_2-10)^2)}-4e^{-0.1((x_1-15)^2+(x_2-5)^2)}$	$0\leq x_i\leq 30, i\in[1,2]$	$f(\vec{x}^*)=-3.98654$, $\vec{x}^*=(15.01369,4.9643)$
f_{24}	Sum of Different Power	$f(x_1,x_2)=\lvert x_2\rvert^3+\lvert x_1\rvert^2$	$-1\leq x_i\leq 1, i\in[1,2]$	$f(\vec{x}^*)=0$, $\vec{x}^*=(0,0)$
f_{25}	Levy	$f(x_1,x_2)=\frac{1}{16}(x_1-1)^2(10\sin^2(\pi(\frac{1}{4}(x_1-1)+1)+1)+1)+\sin^2(\pi(\frac{x_1-1}{4}+1))+\frac{1}{16}(x_2-1)^2(\sin^2(2\pi(\frac{x_2-1}{4}+1))+1)$	$-10\leq x_i\leq 10, i\in[1,2]$	$f(\vec{x}^*)=0$, $\vec{x}^*=(1,1)$
f_{26}	Dejong	$a=$ [[−32, −16, 0, 16, 32, −32, −16, 0, 16, 32, −32, −16, 0, 16, 32, − 32, −16, 0, 16, 32, −32, −16, 0, 16, 32], [−32, −16, 0, 16, 32, −32, −16, 0, 16, 32, −32, −16, 0, 16, 32, −32, −16, 0, 16, 32, −32, −16, 0, 16, 32]] $f(x_1,x_2)=(0.002+\sum_{i=1}^{25}\frac{1}{i+(x_1-a_{1i})^6+(x_2-a_{2i})^6})^{-1}$	$-65.536\leq x_i\leq 65.536$, $i\in[1,2]$	$f(\vec{x}^*)=0.73008956798374342$, $\vec{x}^*=(-31.97855,-31.97855)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{27}	Langermann	$A = [[3.0, 5.0, 2.0, 1.0, 7.0]], [5.0, 2.0, 1.0, 4.0, 9.0]]$ $c = [3.0, 5.0, 2.0, 1.0, 7.0]$ $f(\vec{x}) = \sum_{i=1}^{2} c_i \exp(-\frac{1}{\pi} \sum_{j=1}^{2} (x_j - A_{ij})^2) \cos(\pi \sum_{j=1}^{d} ((x_j - A_{ij})^2))$	$0 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = -5.1197918057700552$, $\vec{x}^* = (6.06958, 8.68035)$
f_{28}	Himmelblau	$f(x_1, x_2) = -\left(x_1 + x_2^2 - 7\right)^2 - \left(x_1^2 + x_2 - 11\right)^2 + 200$	$0 \leq x_i \leq 6, i \in [1, 2]$	$f(\vec{x}^*) = -1986$, $\vec{x}^* = (6, 6)$
f_{29}	Sum squares	$f(x_1, x_2) = x_1^2 + 2x_2^2$	$-10 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0, 0)$
f_{30}	Schaffer2	$f(x_1, x_2) = \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{(0.001(x_1^2 - x_2^2) + 1)^2} + 0.5$	$-100 \leq x_i \leq 100, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (-0.231665, 0.232741)$
f_{31}	Easom	$f(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2$	$-100 \leq x_i \leq 100, i \in [1, 2]$	$f(\vec{x}^*) = -1, \vec{x}^* = (\pi, \pi)$
f_{32}	Matyas	$f(x_1, x_2) = 0.26\left(x_1^2 + x_2^2\right) - 0.48 x_1 x_2$	$-10 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0, 0)$
f_{33}	Cross in Tray	$f(x_1, x_2) = -0.0001(\lvert e^{\lvert 100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi} \rvert} \sin(x_1)\sin(x_2)\rvert + 1)^{0.1}$	$-10 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = -2.06261$, $\vec{x}^* = (1.3492, 1.3491)$
f_{34}	Bukin	$f(x_1, x_2) = 0.01\lvert x_1 + 10\rvert + 100\sqrt{\lvert x_2 - 0.01 x_1^2 \rvert}$	$-15 \leq x_i \leq 3, i \in [1, 2]$	$f(^*) = 0$, $\vec{x}^* = (-10, 1)$
f_{35}	Schaffer4	$f(x_1, x_2) = \frac{\cos(\sin(\lvert x_1^2 - x_2^2 \rvert)) - 0.5}{(0.001(x_1^2 + x_2^2) + 1)^2} + 0.5$	$-100 \leq x_i \leq 100, i \in [1, 2]$	$f(\vec{x}^*) = 0.500092$, $\vec{x}^* = (-99.99634, -99.8942)$
f_{36}	Equal peaks	$f(x_1, x_2) = \sin^2(x_2) + \cos^2(x_1)$	$0 \leq x_i \leq 5, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1.5708, 0)$
f_{37}	Ackley2	$f(x_1, x_2) = -200 e^{-0.02\sqrt{x_1^2 + x_2^2}}$	$-32 \leq x_i \leq 32, i \in [1, 2]$	$f(\vec{x}^*) = -200$, $\vec{x}^* = (0, 0)$
f_{38}	Ackley3	$f(x_1, x_2) = 5e^{\sin(3x_2) + \cos(3x_1)} - 200 e^{-0.02\sqrt{x_1^2 + x_2^2}}$	$-32 \leq x_i \leq 32, i \in [1, 2]$	$f(\vec{x}^*) = -195.629$, $\vec{x}^* = (-0.682577, -0.360702)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{39}	Ackley4	$f(x_1,x_2) = 0.818731\sqrt{x_1^2+x_2^2} + 3(\sin(2x_2)+\cos(2x_1))$	$-35 \leq x_i \leq 35, i \in [1,2]$	$f(\vec{x}^*) = -4.5901016341586682$, $\vec{x}^* = (-1.50962, -0.75487)$
f_{40}	Adjiman	$f(x_1,x_2) = \sin(x_2)\cos(x_1) - \frac{x_1}{x_2^2+1}$	$-1 \leq x_i \leq 2, i \in [1,2]$	$f(\vec{x}^*) = -2.02181$, $\vec{x}^* = (2, 0.10578)$
f_{41}	Alpine1	$f(\vec{x}) = \sum_{i=1}^{D} \lvert x_i sin(x_i) + 0.1x_i \rvert$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{42}	Alpine2	$f(\vec{x}) = \prod_{i=1}^{D} \sqrt{x_i} sin(x)$	$0 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = -6.1295$, $\vec{x}^* = (7.91705, 4.81584)$
f_{43}	Bartels	$f(x_1,x_2) = \lvert x_1^2 + x_2^2 + x_1 * x_2 \rvert + \lvert sin(x_1) \rvert + \lvert cos(x_2) \rvert$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 1$, $\vec{x}^* = (0,0)$
f_{44}	Bigg exp2	$f(x_1,x_2) = \sum_{i=1}^{10} (\exp(-0.1*i*x_1) - 5*\exp(-0.1*i*x_2) - \exp(-0.1*i) - 5*\exp(10*0.1*i))^2$	$0 \leq x_i \leq 20, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1,10)$
f_{45}	Bird	$f(x_1,x_2) = \sin(x_1)*\exp((1-\cos(x_2))^2) + \cos(x_2)*\exp((1-\sin(x_1))^2) + (x_1-x_2)^2$	$-2\pi \leq x_i \leq 2\pi, i \in [1,2]$	$f(\vec{x}^*) = -106.764537$, $\vec{x}^* = (4.70104, 3.15294)$
f_{46}	Bohachevsky2	$f(x_1,x_2) = x_1^2 + 2*x_2^2 - 0.3*\cos(3*\pi*x_1)*0.4*\cos(4*\pi*x_2) + 0.3$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{47}	Bohachevsky3	$f(x_1,x_2) = x_1^2 + 2*x_2^2 - 0.3*\cos(3*\pi*x_1 + 4*\pi*x_2) + 0.3$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{48}	Branin Rcos	$f(x_1,x_2) = (x_2 - (5.1*x_1^2)/(4*\pi^2) + (5*x_1)/\pi - 6)^2 + 10*(1-1/(8*\pi))\cos(x_1) + 10$	$-5 \leq x_i \leq 15, i \in [1,2]$	$f(\vec{x}^*) = 0.3978873$, $\vec{x}^* = (3.14159, 2.275)$
f_{49}	Branin Rcos2	$f(x_1,x_2) = (x_2 - (5.1*x_1^2)/(4*\pi^2) + (5*x_1)/\pi - 6)^2 + 10*(1-1/(8*\pi))\cos[x_1)\cos(x_2)\log(x_1^2+x_2^2+1) + 10$	$-5 \leq x_i \leq 15, i \in [1,2]$	$f(\vec{x}^*) = -39.195653917977752$, $\vec{x}^* = (-3.1721, 12.58567)$
f_{50}	Brent	$f(x_1,x_2) = (x_1+10)^2 + (x_2+10)^2 + \exp(-x_1^2 - x_2^2)$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{51}	Brown	$f(x_1,x_2) = (x_1^2)^{(x_2^2+1)} + (x_2^2)^{(x_1^2+1)}$	$-1 \leq x_i \leq 4, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{52}	Bukin2	$f(x_1,x_2) = 100*(x_2 - 0.01x_1^2 + 1) + 0.01(x_1+10)^2$	$-15 \leq x_i \leq 3, i \in [1,2]$	$f(\vec{x}^*) = -1624.75$, $\vec{x}^* = (-15, -15)$
f_{53}	Bukin4	$f(x_1,x_2) = 100x_2^2 + 0.01*\lvert x_1 + 10 \rvert$	$-15 \leq x_i \leq 3, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (-10, 0)$
f_{54}	Three Hump Camel function	$f(x_1,x_2) = 2x_1^2 - 1.05x_1^4 + x_1^6/6 + x_1x_2 + x_2^2$	$-5 \leq x_i \leq 5, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{55}	Six Hump Camel function	$f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^2/3)x_1^2 + x_1x_1 + (4x_2^2 - 4)x_2^2$	$-5 \leq x_i \leq 5, i \in [1, 2]$	$f(\vec{x}^*) = -1.0316$, $\vec{x}^* = (-0.0898, 0.7126)$
f_{56}	Chen Bird	$f(x_1, x_2) = -0.001/(0.001^2 + (x_1 - 0.4x_2 - 0.1)^2) - 0.001/(0.001^2 + (2x_1 + x_2 - 1.5)^2)$	$-500 \leq x_i \leq 500, i \in [1, 2]$	$f(\vec{x}^*) = -1000$, $\vec{x}^* = (0.149371, 0.123427)$
f_{57}	Chenv	$f(x_1, x_2) = -(0.001/(0.001^2 + (x_1^2 + x_2^2 - 1)^2)) - 0.001/(0.001^2 + (x_1^2 + x_2^2 - 0.5)^2) - 0.001/(0.001^2 + (x_1^2 - x_2^2)^2)$	$-500 \leq x_i \leq 500, i \in [1, 2]$	$f(\vec{x}^*) = -2000$, $\vec{x}^* = (-0.5, -0.5)$
f_{58}	Chichinadze	$f(x_1, x_2) = x_1^2 - 12x_1 + 11 + 10\cos((\pi x_1)/2) + 8\sin((5\pi x_1)/2) - (1/5)^{0.5}\exp(-0.5(x_2 - 0.5)^2)$	$-30 \leq x_i \leq 30, i \in [1, 2]$	$f(\vec{x}^*) = -42.49717342349103$, $\vec{x}^* = (6.18987, 0.75477)$
f_{59}	Chung Reynolds	$f(\vec{x}) = (\sum_{i=1}^{2} x_i^2)^2$	$-100 \leq x_i \leq 100, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0, 0)$
f_{60}	Cosine mixture	$f(\vec{x}) = -0.1\sum_{i=1}^{2}\cos(5\pi x_i) - \sum_{i=1}^{2} x_i^2$	$-1 \leq x_i \leq 1, i \in [1, 2]$	$f(\vec{x}^*) = -1.7987686839243868$, $\vec{x}^* = (0.9995, 0.99988)$
f_{61}	Csendes	$f(\vec{x}) = \sum_{i=1}^{2} x_i^6 * (2 + \sin(x_i))$	$-1 \leq x_i \leq 1, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0, 0)$
f_{62}	Cube	$f(x_1, x_2) = 100 * (x_2 - x_1^3)^2 + (1 - x_1)^2$	$-10 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1, 1)$
f_{63}	Damavandi	$f(x_1, x_2) = (1 - \lvert(\sin(\pi(x_1 - 2))\sin(\pi(x_2 - 2)))/(\pi^2(x_1 - 2)(x_[2 - 2))\rvert^5)/(2 + (x_1 - 7)^2 + 2(x_2 - 7)^2)$	$0 \leq x_i \leq 14, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (2, 2)$
f_{64}	Deckkers Aarts	$f(x_1, x_2) = 10^5 * x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^-5(x_1^2 + x_2^2)^4$	$-20 \leq x_i \leq 20, i \in [1, 2]$	$f(\vec{x}^*) = -24771.093749999996$, $\vec{x}^* = (0, -15)$
f_{65}	El Attar Vidyasagar Dutta	$f(x_1, x_2) = (x_1^2 + x_2 - 10)^2 + (x_1 + x_2^2 - 7)^2 + (x_1^2 + x_2^3 - 1)^2$	$-500 \leq x_i \leq 500, i \in [1, 2]$	$f(\vec{x}^*) = 1.7127803597192561$, $\vec{x}^* = (3.40919, -2.17143)$
f_{66}	Egg crate	$f(x_1, x_2) = x_1^2 + x_2^2 + 25(\sin(x_1)^2 + \sin(x_2))$	$-5 \leq x_i \leq 5, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0, 0)$
f_{67}	Exponential	$f(\vec{x}) = -\exp(-0.5 * \sum_{i=1}^{D} x_i^2)$	$-1 \leq x_i \leq 1, i \in [1, 2]$	$f(\vec{x}^*) = 1$, $\vec{x}^* = (0, 0)$
f_{68}	Exp2	$f(x_1, x_2) = \sum_{i=0}^{9}(\exp(-(ix_1)/10) - 5\exp(-(ix_2)/10) - \exp[-i/10] + 5\exp(-i)^2$	$0 \leq x_i \leq 20, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1, 10)$
f_{69}	Freudenstein roth	$f(x_1, x_2) = (x_1 - 13 + ((5 - x_2)x_2 - 2)x_2)^2 + (x_1 - 29 + ((x_2 + 1)x_2 - 14)x_2)^2$	$-10 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (5, 4)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{70}	Giunta	$f(\vec{x}) = 0.6 + \sum_{i=1}^{D}(\sin(16/15x_i - 1) + \sin(16/15x_i - 1)^2 + 1/50\sin(4(16/15x_i - 1)))$	$-1 \leq x_i \leq 1, i \in [1,2]$	$f(\vec{x}^*) = 0.0644704$, $\vec{x}^* = (0.46732, 0.46732)$
f_{71}	Hansen	$f(x_1,x_2) = \sum_{i=0}^{4}(i+1)\cos(ix_1+i+1)\sum_{j=0}^{4}(j+1)\cos((j+2)x_2+j+1$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = -176.542$, $\vec{x}^* = (-7.589893, -7.708314)$
f_{72}	Hosaki	$f(x_1,x_2) = (1 - 8x_1 + 7x_1^2 - 7/3x_1^3 + 1/4x_1^4)x_2^2\exp(-x_2)$	$0 \leq x_i \leq 6, i \in [1,2]$	$f(\vec{x}^*) = -2.3458$, $\vec{x}^* = (4,2)$
f_{73}	Jennrich Sampson	$f(x_1,x_2) = \sum_{i=1}^{10}(2 + 2i - (\exp(ix_1) + \exp(ix_2)))^2$	$-1 \leq x_i \leq 1, i \in [1,2]$	$f(\vec{x}^*) = 124.96218236181409$, $\vec{x}^* = (0.257825, 0.257825)$
f_{74}	Keane	$f(x_1,x_2) = (\sin(x_1 - x_2)^2\sin(x_1 + x_2)^2)/\sqrt{(x_1^2 + x_2^2)}$	$0 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 2.4590189858452324e-36$, $\vec{x}^* = (-8.69395e-9, 8.69394e-9)$
f_{75}	Leon	$f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$-1.2 \leq x_i \leq 1.2, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1,1)$
f_{76}	Mccormick	$f(x_1,x_2) = \sin(x_1+x_2)+(x_1+x_2)^2-(3/2)*x_1+(5/2)*x_2+1$	$-3 \leq x_i \leq 4, i \in [1,2]$	$f(\vec{x}^*) = -11.06537266363643$, $\vec{x}^* = (3.26783, -3.0)$
f_{77}	Mishra3	$f(x_1,x_2) = \sqrt{\lvert\cos(\sqrt{\lvert x_1^2 + x_2^2\rvert})\rvert} + 0.01(x_1 + x_2)$	$-1 \leq x_i \leq 1, i \in [1,2]$	$f(\vec{x}^*) = 0.3748970685702472$, $\vec{x}^* = (3.26783, -3.0)$
f_{78}	Mishra4	$f(x_1,x_2) = \sqrt{(\lvert\cos(\sqrt{(\lvert x_1^2 + x_2^2\rvert)})\rvert)} + 0.01(x_1 + x_2)$	$-1 \leq x_i \leq 1, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{79}	Mishra5	$f(x_1,x_2) = (\sin(\cos(x_1 + \cos(x_2)))^2 + \cos(\sin(x_1 + \sin(x_2)))^2)^2 + 0.01(x_1 + x_2)^2$	$-1 \leq x_i \leq 1, i \in [1,2]$	$f(\vec{x}^*) = -0.01864212603865322$, $\vec{x}^* = (-0.86535, -1.0)$
f_{80}	Mishra6	$f(x_1,x_2) = -\log(\sin(\cos(x_1) + \cos(x_2))^2 - \cos(\sin(x_1) + \sin(x_2))^2 + x_1)^2 + 0.01((x_1 - 1)^2 + (x_2 - 1)^2)$	$1 \leq x_i \leq 6, i \in [1,2]$	$f(\vec{x}^*) = -0.809819$, $\vec{x}^* = (2,2)$
f_{81}	Mishra8	$f(x_1,x_2) = 0.001(\lvert x_1^10 - 20x_1^9 + 180x_1^8 - 960x_1^7 + 3360x_1^6 - 8064x_1^5 + 1334x_1^4 - 15360x_1^3 + 11520x_1^2 - 5120x_1 + 2624\rvert\rvert x_2^4 + 12x_2^3 + 54x_2^2 + 108x_2 + 81\rvert)^2$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (2,-3)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{82}	Pen Holder	$f(x_1,x_2) = -\exp(-\lvert\cos(x_1)\cos(x_2)\exp(\lvert 1 - (x_1^2 + x_2^2)^0.5/\pi\rvert)\rvert^{-}1)$	$-11 \leq x_i \leq 11, i \in [1,2]$	$f(\vec{x}^*) = -0.963535$, $\vec{x}^* = (9.646168, 9.646168)$
f_{83}	Pathological	$f(x_1,x_2) = 0.5 + (\sin(\sqrt{100x_1 + x_2^2})^2 - 0.5)/(1 + 0.001 * (x_1^2 - 2x_1x_2 + x_2^2)^2), i, 1, 1]$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{84}	Periodic	$f(x_1,x_2) = 1 + \sin(x_1)^2 + \sin(x_2)^2 - 0.1\exp[-(x_1^2 + x_2^2)]$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0.9$, $\vec{x}^* = (0,0)$
f_{85}	Powell sum	$f(\vec{x}) = \sum_{i=1}^{2} \lvert x_i\rvert^{(i+1)}$	$-1 \leq x_i \leq 1, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{86}	Price1	$f(x_1,x_2) = (\lvert x_1\rvert - 5)^2 + (\lvert x_2\rvert - 5)^2$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (-5,5)$
f_{87}	Price2	$f(x_1,x_2) = 1 + \sin(x_1)^2 + \sin(x_2)^2 - 0.1\exp(-x_1^2 - x_2^2\rvert$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0.9$, $\vec{x}^* = (0,0)$
f_{88}	Price3	$f(x_1,x_2) = 100(x_2 - x_1^2)^2 + 6(6.4(x_2 - 0.5)^2 - x_1 - 0.6)^2$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 1.43791935893e-11$, $\vec{x}^* = (0.341308, 0.116491)$
f_{89}	Price4	$f(x_1,x_2) = (2x_1^3x_2 - x_2^3)^2 + (6x_1 - x_2^2 + x_2)^2$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{90}	Qing	$f(\vec{x}) = \sum_{i=1}^{2} (x_i^2 - 1)^2$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{91}	Quadratic	$f(x_1,x_2) = -3803.84 - 138.08x_1 - 232.92x_2 + 128.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2$	$-1 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = -3873.7241821830326$, $\vec{x}^* = (0.19388, 0.48513)$
f_{92}	Quartic	$f(\vec{x}) = \sum_{i=1}^{2} ix_i^2 + rand[0,1)$	$-1.28 \leq x_i \leq 1.28, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{93}	Quintic	$f(\vec{x}) = \sum_{i=1}^{2} \lvert x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4\rvert$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (2,2)$
f_{94}	Rosenbrock modified	$f(x_1,x_2) = 74 + 100(x_2 - x_1^2)^2 + (1 - x_1)^2 - 400\exp(-((x_1 + 1)^2 + (x_2 + 1)^2)/0.1)$	$-2 \leq x_i \leq 2, i \in [1,2]$	$f(\vec{x}^*) = 74$, $\vec{x}^* = (1,1)$
f_{95}	Rotated ellipse	$f(x_1,x_2) = 7x_1^2 - 6\sqrt{3}x_1x_2 + 13x_2^2$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{96}	Rotated ellipse2	$f(x_1,x_2) = x_1^2 - x_1x_2 + x_2^2$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{97}	Rump	$f(x_1,x_2) = (333.75 - x_1^2)x_2^6 + x_1^2(11x_1^2x_2^2 - 121x_2^4 - 2) + 5.5x_2^8 + x_1/(2)$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = -2.44021e18$, $\vec{x}^* = (500, 180)$
f_{98}	Salomon	$f(x_1,x_2) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{2} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{99}	Sargan	$f(\vec{x}) = \sum_{i=1}^{2} 2 * (x_i^2 + 0.4x_ix_2)$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{100}	Schaffer3	$f(x_1,x_2) = 0.5 + (\sin(\cos(\lvert x_1^2 - x_2^2\rvert))^2 - 0.5)/(1 + 0.001(x_1^2 + x_2^2)^2)$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = 0.00123013247589431$, $\vec{x}^* = (0, 1.253115)$
f_{101}	Schumer Steiglitz	$f(\vec{x}) = \sum_{i=1}^{2} x_i^4$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{102}	Schwefel24	$f(\vec{x}) = \sum_{i=1}^{2}(x_i - 1)^2 + (x_1 - x_i^2)^2$	$0 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1,1)$
f_{103}	Schwefel222	$f(\vec{x}) = \sum_{i=1}^{2} \lvert x_i\rvert + \prod_{i=1}^{2} \lvert x_i\rvert$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{104}	Schwefel236	$f(x_1,x_2) = -x_1x_2(72 - 2x_1 - 2x_2)$	$0 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = -3456$, $\vec{x}^* = (12,12)$
f_{105}	Streched v sine wave	$f(x_1,x_2) = (x_2^2 + x_1^2)^{0.25}(\sin(50(x_2^2 + x_1)^{0}.1)^2 + 0.1)$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{106}	Testtube holder	$f(x_1,x_2) = -4(\sin(x_1)\cos(x_2)\exp(\lvert \cos((x_1^2 + x_2^2)/200)\rvert))$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = -10.872300$, $\vec{x}^* = (\pi/2.0, 0)$
f_{107}	Trecanni	$f(x_1,x_2) = x_1^4 - 4x_1^3 + 4x_1x_2^2$	$-5 \leq x_i \leq 5, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{108}	Trefethen	$f(x_1,x_2) = \exp(50\sin(x_1)) + \sin(60\exp(x_2)) + \sin(70\sin(x_1)) + \sin(\sin(80x_2)) - \sin(10(x_1+x_2)) + 1.0/4.0(x_1^2 + x_2^2)$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = -3.30686865$, $\vec{x}^* = (-0.024403, 0.210612)$
f_{109}	Trigonometric	$f(\vec{x}) = \sum_{i=1}^{2}(2 - \sum_{j=1}^{2}(\cos(x_j)) + i(1 - \cos(x_i) - \sin(x_i)))^2$	$0 \leq x_i \leq \pi, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
f_{110}	Trigonometric 2	$f(\vec{x}) = 1 + \sum_{i=1}^{2} 8(\sin(7(x_i - 0.9)^2))^2 + 6(\sin(14(x_1 - 0.9)))^2 + (x_i - 0.9)^2$	$-500 \leq x_i \leq 500, i \in [1,2]$	$f(\vec{x}^*) = 1$, $\vec{x}^* = (0.9, 0.9)$

Conflict of Interest: None of the authors present any conflicts of interest whatsoever.

References

Altenberg, L. (1997). Fitness distance correlation analysis: An instructive counterexample. In T. Back (Ed.), *Icga* (p. 57-64). Morgan Kaufmann.

Auger, A., Hansen, N., Heidrich-Meisner, V., Mersmann, O., Posik, P., & Preuss, M. (2012). Gecco 2012 workshop on black-box optimization benchmarking (bbob). In *Gecco 2012: Genetic and evolutionary computation conference companion.* New York, NY, USA: ACM. Retrieved from `http://coco.gforge.inria.fr/doku.php?id=bbob-2012`

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32. Retrieved from `http://dx.doi.org/10.1023/A%3A1010933404324` doi: 10.1023/A:1010933404324

Caamaño, P., Bellas, F., Becerra, J. A., & Duro, R. J. (2013). Evolutionary algorithm characterization in real parameter optimization problems. *Appl. Soft Comput.*, *13*(4), 1902-1921. Retrieved from `http://dblp.uni-trier.de/db/journals/asc/asc13.html#CaamanoBBD13`

Caamaño, P., Prieto, A., Becerra, J. A., Bellas, F., & Duro, R. J. (2010). Real-valued multimodal fitness landscape characterization for evolution. In *Proceedings of the 17th international conference on neural information processing: Theory and algorithms - volume part i* (pp. 567–574). Berlin, Heidelberg: Springer-Verlag. Retrieved from `http://dl.acm.org/citation.cfm?id=1939659.1939733`

Chan, K. Y., Aydin, M. E., & Fogarty, T. C. (2003). An epistasis measure based on the analysis of variance for the real-coded representation in genetic algorithms. In *Ieee congress on evolutionary computation* (p. 297-304). IEEE. Retrieved from `http://dblp.uni-trier.de/db/conf/cec/cec2003-1.html#ChanAF03`

Chen, Y., Hu, J., Hirasawa, K., & Yu, S. (2008). Solving deceptive problems using a genetic algorithm with reserve selection. In *Evolutionary computation, 2008. cec 2008. (ieee world congress on computational intelligence). ieee congress on* (p. 884-889). doi: 10.1109/CEC.2008.4630900

Davidor, Y. (1990). Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, *4*, 369-383.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009, November). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, *11*(1), 10–18. Retrieved from `http://doi.acm.org/10.1145/1656274.1656278` doi: 10.1145/1656274.1656278

Hansen, N., Auger, A., Finck, S., & Ros, R. (2010, September). *Real-parameter black-box optimization benchmarking 2010: Experimental setup* (Tech. Rep. No. RR-7215). Paris, France: INRIA.

He, J., Reeves, C., Witt, C., & Yao, X. (2007, December). A note on problem difficulty measures in black-box optimization: Classification,

realizations and predictability. *Evol. Comput.*, *15*(4), 435–443. Retrieved from http://dx.doi.org/10.1162/evco.2007.15.4.435 doi: 10.1162/evco.2007.15.4.435

Herrera, F., Lozano, M., & Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, *12*, 265–319.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence.* Cambridge, MA, USA: MIT Press.

Horn, J., & Goldberg, D. E. (1995). Genetic algorithm difficulty and the modality of the fitness landscapes. In L. D. Whitley & M. D. Vose (Eds.), *Foundations of genetic algorithms workshop, 3* (p. 243-269). Morgan Kaufmann.

Jamil, M., & Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *IJMNO*, *4*(2), 150-194. Retrieved from http://dblp.uni-trier.de/db/journals/ijmno/ijmno4.html#JamilY13

Jones, T. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search* (Doctoral dissertation, University of New Mexico). Retrieved from http://jones.tc/research/phd.html

Jones, T., & Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th international conference on genetic algorithms* (pp. 184–192). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=645514.657929

Kauffman, S. A., & Johnsen, S. (1991). Coevolution to the edge of chaos: Coupled fitness landscapes, poised states, and coevolutionary avalanches. *Journal of Theoretical Biology*, *149*(4), 467–505.

Kerschke, P., Preuss, M., Hernández, C., Schütze, O., Sun, J., Grimme, C., ... Trautmann, H. (2014). Cell mapping techniques for exploratory landscape analysis. In A. Tantar et al. (Eds.), *Evolve — a bridge between probability, set oriented numerics, and evolutionary computation v* (Vol. 288, pp. 115–131). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-07494-8_9 (Publication status: Published) doi: 10.1007/978-3-319-07494-8_9

Kimura, M. (1983). *The neutral theory of molecular evolution.* Cambridge: Cambridge University Press.

Lobo, J., Miller, J. H., & Fontana, W. (2004). *Neutrality in Technological Landscapes* (Tech. Rep.). working paper, Santa Fe Institute, Santa Fe.

López, E. G., & Poli, R. (2006). Some steps towards understanding how neutrality affects evolutionary search. In T. P. Runarsson, H.-G. Beyer, E. K. Burke, J. J. M. Guervós, L. D. Whitley, & X. Yao (Eds.), *Ppsn* (Vol. 4193, p. 778-787). Springer. Retrieved from http://dblp.uni-trier.de/db/conf/ppsn/ppsn2006.html#LopezP06

Malan, K., & Engelbrecht, A. P. (2009). Quantifying ruggedness of continuous landscapes using entropy. In *Ieee congress on evolutionary computation*

(p. 1440-1447). IEEE. Retrieved from http://dblp.uni-trier.de/db/conf/cec/cec2009.html#MalanE09

Malan, K. M., & Engelbrecht, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, *241*(0), 148 - 163. Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025513003125 doi: http://dx.doi.org/10.1016/j.ins.2013.04.015

Mario Graff and Riccardo Poli. (2010, October). Practical performance models of algorithms in evolutionary program induction and other domains. *Artificial Intelligence*, *174*(15), 1254–1276. Retrieved 2011-07-02, from http://www.sciencedirect.com/science/article/pii/S000437021000127X doi: 16/j.artint.2010.07.005

Merkuryeva G., B. V. (2011). Benchmark fitness landscape analysis. *International Journal of Simulation Systems, , Science and Technology*, *12*(2), 38–45.

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., & Rudolph, G. (2011). Exploratory landscape analysis. In *Proceedings of the 13th annual conference on genetic and evolutionary computation* (pp. 829–836). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2001576.2001690 doi: 10.1145/2001576.2001690

Muñoz, M. A., Kirley, M., & Halgamuge, S. K. (2015). Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE Trans. Evolutionary Computation*, *19*(1), 74–87. Retrieved from http://dx.doi.org/10.1109/TEVC.2014.2302006 doi: 10.1109/TEVC.2014.2302006

Naudts, B., & Kallel, L. (2000a). A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *4*(1), 1–15.

Naudts, B., & Kallel, L. (2000b, Apr). A comparison of predictive measures of problem difficulty in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, *4*(1), 1-15. doi: 10.1109/4235.843491

Ochoa, G., Tomassini, M., Vérel, S., & Darabos, C. (2008). A study of nk landscapes' basins and local optima networks. In C. Ryan & M. Keijzer (Eds.), *Gecco* (p. 555-562). ACM. Retrieved from http://dblp.uni-trier.de/db/conf/gecco/gecco2008.html#OchoaTVD08

Pitzer, E., & Affenzeller, M. (2012). A comprehensive survey on fitness landscape analysis. In J. C. Fodor, R. Klempous, & C. P. S. Araujo (Eds.), *Recent advances in intelligent engineering systems* (Vol. 378, p. 161-191). Springer. Retrieved from http://dblp.uni-trier.de/db/series/sci/sci378.html#PitzerA12

Pitzer, E., Affenzeller, M., & Beham, A. (2010, Sept). A closer look down the basins of attraction. In *Computational intelligence (ukci), 2010 uk workshop on* (p. 1-6). doi: 10.1109/UKCI.2010.5625595

Reeves, C. R., & Wright, C. C. (1995). Epistasis in genetic algorithms: An experimental design perspective. In *Proc. of the 6th international conference on genetic algorithms, (pp 217–224* (pp. 217–224). Morgan Kauf-

mann.

Reidys, C. M., & Stadler, P. F. (2002). Combinatorial landscapes. *SIAM Rev.*, *44*(1), 3–54 (electron.

Rodriguez-Maya, N., Graff, M., & Flores, J. (2014). Performance classification of genetic algorithms on continuous optimization problems. In A. Gelbukh, F. Espinoza, & S. Galicia-Haro (Eds.), *Nature-inspired computation and machine learning* (Vol. 8857, p. 1-12). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-13650-9_1 doi: 10.1007/978-3-319-13650-9_1

Smith, T., Philippides, A., Husbands, P., & O'Shea, M. (2002). Neutrality and ruggedness in robot landscapes. In *Evolutionary computation, 2002. cec '02. proceedings of the 2002 congress on* (Vol. 2, p. 1348-1353). doi: 10.1109/CEC.2002.1004439

Trujillo, L., Martínez, Y., Galván López, E., & Legrand, P. (2012). A comparative study of an evolvability indicator and a predictor of expected performance for genetic programming. In *Proceedings of the fourteenth international conference on genetic and evolutionary computation conference companion* (pp. 1489–1490). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2330784.2331006 doi: 10.1145/2330784.2331006

Trujillo, L., Martínez, Y., López, E. G., & Legrand, P. (2011). Predicting problem difficulty for genetic programming applied to data classification. In *Gecco* (p. 1355-1362).

Vanneschi, L. (2004). *Theory and practice for efficient genetic programming* (Doctoral dissertation, Faculty of Sciences, University of Lausanne, Switzerland). Retrieved from http://old.disco.unimib.it/Vanneschi/thesis_vanneschi.pdf

Vanneschi, L., Clergue, M., Collard, P., Tomassini, M., & Vérel, S. (2004). Fitness clouds and problem hardness in genetic programming. In K. Deb (Ed.), *Genetic and evolutionary computation - gecco 2004* (Vol. 3103, p. 690-701). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-24855-2_76 doi: 10.1007/978-3-540-24855-2_76

Vanneschi, L., & Tomassini, M. (2002, 8 July). A study on fitness distance correlation and problem difficulty for genetic programming. In S. Luke, C. Ryan, & U.-M. O'Reilly (Eds.), *Graduate student workshop* (pp. 307–310). New York: AAAI. Retrieved from http://personal.disco.unimib.it/Vanneschi/GECCO_2002_PHD_WORKSHOP.pdf

Vanneschi, L., Tomassini, M., Collard, P., & Clergue, M. (2005). A survey of problem difficulty in genetic programming. In S. Bandini & S. Manzoni (Eds.), *Ai*ia 2005: Advances in artificial intelligence* (Vol. 3673, p. 66-77). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/11558590_7 doi: 10.1007/11558590_7

Vanneschi, L., Tomassini, M., Collard, P., & Vérel, S. (2006). Negative Slope Coefficient: A Measure to Characterize Genetic Programming Fitness Landscapes. In *Genetic programming, 9th european conference, eurogp*

2006, budapest, hungary, april 10-12, 2006, proceedings (pp. 178–189). Springer. Retrieved from http://dx.doi.org/10.1007/11729976_16 doi: 10.1007/11729976_16

Vanneschi, L., Tomassini, M., Pirola, Y., Verel, S., & Mauri, G. (2006). A quantitative study of neutrality in gp boolean landscapes. In *Proceedings of the genetic and evolutionary computation conference, gecco'06* (pp. 895–902). ACM Press.

Vanneschi, L., Valsecchi, A., & Poli, R. (2009). Limitations of the fitness-proportional negative slope coefficient as a difficulty measure. In *Proceedings of the 11th annual conference on genetic and evolutionary computation* (pp. 1877–1878). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1569901.1570212 doi: 10.1145/1569901.1570212

Vanneschi, L., Verel, S., Tomassini, M., & Collard, P. (2009). Nk landscapes difficulty and negative slope coefficient: How sampling influences the results. In M. Giacobini et al. (Eds.), *Applications of evolutionary computing* (Vol. 5484, p. 645-654). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-01129-0_74 doi: 10.1007/978-3-642-01129-0_74

Vassilev, V. K., Miller, J. F., & Fogarty, T. C. (1999, 6-9 July). Digital circuit evolution and fitness landscapes. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, & A. Zalzala (Eds.), *Proceedings of the congress on evolutionary computation* (Vol. 2). Mayflower Hotel, Washington D.C., USA: IEEE Press. doi: doi:10.1109/CEC.1999.782595

Volke, S., Bin, S., Zeckzer, D., Middendorf, M., & Scheuermann, G. (2014). Fitness landscapes: From evolutionary biology to evolutionary computation. In *Recent advances in the theory and application of fitness landscapes* (Hendrik Richter and Andries Engelbrecht ed., Vol. 6, p. 487 - 507). Berlin, Heidelberg: Springer. Retrieved from http://dx.doi.org/10.1007/978-3-642-41888-4_17 doi: 10.1007/978-3-642-41888-4_17

Vérel, S., Collard, P., & Clergue, M. (2007). Where are bottlenecks in nk fitness landscapes? *CoRR*, *abs/0707.0641*. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr0707.html#abs-0707-0641

Weise, T. (2009). *Global Optimization Algorithms – Theory and Application*. it-weise.de (self-published): Germany. Retrieved from http://www.it-weise.de/projects/book.pdf

Wolpert, D. H., & Macready, W. (1995). *No free lunch theorems for search* (Tech. Rep. No. SFI-TR-95-01-010). Santa Fe, NM: The Santa Fe Institute.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Wright, A. H. (1991). Genetic algorithms for real parameter optimization. In *Foundations of genetic algorithms* (pp. 205–218). Morgan Kaufmann.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of*

Genetics, *1*, 356-66.

Xin, B., Chen, J., & Pan, F. (2009). Problem difficulty analysis for particle swarm optimization: Deception and modality. In *Proceedings of the first acm/sigevo summit on genetic and evolutionary computation* (pp. 623–630). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1543834.1543919` doi: 10.1145/1543834.1543919