UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica

CHARACTERIZING THE PERFORMANCE OF EVOLUTIONARY ALGORITHMS TO SOLVE CONTINUOUS OPTIMIZATION PROBLEMS

TESIS

Que para obtener el grado de:
DOCTOR EN CIENCIAS EN INGENIERÍA ELÉCTRICA

Presenta:
NOEL ENRIQUE RODRÍGUEZ MAYA

Director de Tesis
Dr. Juan José Flores Romero

Co-Director de Tesis
Dr. Mario Graff Guerrero

Morelia, Michoacán, Julio 2016

RESUMEN

En el campo de Algoritmos Evolutivos (AE), ¿qué constituye un problema de optimización difícil?, ¿por qué ciertos problemas de optimización son más difíciles de resolver? Las respuestas a estas preguntas pueden proveer información importante, por ejemplo, la predicción del éxito de los AE. Para predecir el éxito de los AE muchos enfoques han sido desarrollados; siendo el estudio de Fitness Landscape (FL) uno de los más exitosos. Básicamente, FL es la forma geométrica representada por la función de costo de los problemas de optimización; el FL puede ser calculado de métricas para medir, por ejemplo, la tasa de rugosidad, neutralidad, cuencas de atracción, entre otras. Fitness Landscape Analysis usa un conjunto de métricas para la caracterización del FL: las características de los problemas se pueden medir por medio de métricas descriptivas, mientras que las métricas dinámicas están más relacionadas a las características del algoritmo. Esta contribución presenta un procedimiento llamado Modelos de Clasificación de Rendimiento (PCM) el cual crea modelos para predecir el rendimiento exhibido por los Algoritmos Genéticos (AG) en la solución de problemas de optimización en dominios continuos. PCM clasifica el rendimiento en dos clases (fácil o difícil). El conjunto de datos tiene características del FL como variables predictoras, y el rendimiento exhibido por el AG como variable objetivo. Los problemas usados en los experimentos son funciones de optimización de referencia. Un producto de PCM, es un procedimiento para Recomendar Tamaño de Población (RPS): dado un problema de optimización, RPS recomienda la población mínima para obtener un nivel eficiente de rendimiento. Este trabajo puede ser fácilmente extendido para usar otras métricas, se puede aplicar a un conjunto diferente de problemas, o usar otro AE. El desarrollo de modelos de rendimiento para otros AE, puede llevar a la solución de una instancia del problema de selección de algoritmo.

Palabras clave: Optimización, Fitness Landscape, Algoritmos Evolutivos, Algoritmos Genéticos, Predicción.

ABSTRACT

In the field of Evolutionary Algorithms (EA), what does constitutes a hard optimization problem?, why certain optimization problems are more difficult to solve? The answers to these questions can provide useful information, e.g., the prediction of the success of EA. In order to predict the success of EA many approaches have been developed; being the study of Fitness Landscape (FL) one of the most successful. Basically, FL is the geometric form depicted by the cost function of optimization problems; the FL can be computed by means of metrics to measure, e.g., the rate of ruggedness, neutrality, basins of attraction, among others. Fitness Landscape Analysis uses a set of metrics for the FL characterization: features of problems can be measured by descriptive metrics, while the dynamic metrics are more related to the features of the algorithm. This contribution presents a procedure called Performance Classification Models (PCM), which creates models to predict the performance exhibited by Genetic Algorithms (GA) in the solution of optimization problems in continuous domains. PCM classifies the performance in two classes (easy and difficult). The dataset has FL features as predictor variables, and the performance exhibited by GA as the target variable. The problems used in experiments are benchmark optimization functions. A product of PCM, is a procedure to Recommend Population Size (RPS): given an optimization problem, RPS recommends the minimal population size to get an acceptable level of performance. This work can be easily extended to use other metrics, it can be applied to different set of problems, or use other EA. Developing performance models for other EA, can lead to the solution of an instance of the algorithm selection problem.

CHARACTERIZING THE PERFORMANCE OF EVOLUTIONARY ALGORITHMS TO SOLVE CONTINUOUS OPTIMIZATION PROBLEMS

TESIS

Que para obtener el grado de

DOCTOR EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Noel Enrique Rodríguez Maya

Juan José Flores Romero

Director de Tesis

Mario Graff Guerrero

Co-Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Julio 2016

CHARACTERISING THE PERFORMANCE OF EVOLUTIONARY ALGORITHMS TO SOLVE CONTINUOUS OPTIMIZATION PROBLEMS

Los Miembros del Jurado de Examen de Grado aprueban la **Tesis de Doctorado en Ciencias en Ingeniería Eléctrica, Opción en Sistemas Computacionales** de ***Noel Rodríguez Maya***

Dr. Jaime Cerda Jacobo
Presidente del Jurado

Dr. Juan José Flores Romero
Director de Tesis

Dr. Mario Graff Guerrero
Co-director

Dr. Félix Calderón Solorio
Vocal

Dr. Sébastien Verel
Revisor Externo (Université du Littoral Cote d¨Opale ULCO, France)

Dr. Félix Calderón Solorio
Jefe de la División de Estudios de Posgrado
de la Facultad de Ingeniería Eléctrica. UMSNH
(Por reconocimiento de firmas)

UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Abril 2016

Dedication

A mi esposa Mónica por el apoyo incondicional,

a mis hijas Valeria y Noelia por su paciencia y comprensión,

a mis padres Ofelia y Ascención por sus consejos y cuidados,

a mis hermanos Maira, Erick y Ericka por estar siempre presentes,

a toda mi familia por su alegría y motivación.

a mis amigos por hacerme ver mis errores.

Acknowledgements

Quiero agradecer primeramente a mi alma máter el Instituto Tecnológico de Zitácuaro por darme la oportunidad de continuar mis estudios de posgrado.

Agradezco al Dr. Juan José Flores Romero por su apoyo y consejos durante los estudios de doctorado, al Dr. Mario Graff Guerrero por sus atinadas observaciones para el desarrollo del presente trabajo y al Dr. Sébastien Verel por su tiempo.

También quiero agradecer a CONACYT por el soporte económico durante todo el periodo que comprendió esta investigación.

Abstract

In the field of Evolutionary Algorithms (EA), what does constitutes a hard optimization problem?, why certain optimization problems are more difficult to solve? The answers to these questions can provide useful information, e.g., the prediction of the success of EA. In order to predict the success of EA many approaches have been developed; being the study of Fitness Landscape (FL) one of the most successful. Basically, FL is the geometric form depicted by the cost function of optimization problems; the FL can be computed by means of metrics to measure, e.g., the rate of ruggedness, neutrality, basins of attraction, among others. Fitness Landscape Analysis uses a set of metrics for the FL characterization: features of problems can be measured by descriptive metrics, while the dynamic metrics are more related to the features of the algorithm. This contribution presents a procedure called *Performance Classification Models* (PCM), which creates models to predict the performance exhibited by Genetic Algorithms (GA) in the solution of optimization problems in continuous domains. PCM classifies the performance in two classes (easy and difficult). The dataset has FL features as predictor variables, and the performance exhibited by GA as the target variable. The problems used in experiments are benchmark optimization functions. A product of PCM, is a procedure to *Recommend Population Size* (RPS): given an optimization problem, RPS recommends the minimal population size to get an acceptable level of performance. This work can be easily extended to use other metrics, it can be applied to different set of problems, or use other EA. Developing performance models for other EA, can lead to the solution of an instance of the algorithm selection problem.

Resumen

En el campo de Algoritmos Evolutivos (AE), ¿qué constituye un problema de optimización difícil?, ¿por qué ciertos problemas de optimización son más difíciles de resolver? Las respuestas a estas preguntas pueden proveer información importante, por ejemplo, la predicción del éxito de los AE. Para predecir el éxito de los AE muchos enfoques han sido desarrollados; siendo el estudio de Fitness Landscape (FL) uno de los más exitosos. Básicamente, FL es la forma geométrica representada por la función de costo de los problemas de optimización; el FL puede ser calculado de metrics para medir, por ejemplo, la tasa de rugosidad, neutralidad, cuencas de atracción, entre otras. Fitness Landscape Analysis usa un conjunto de métricas para la caracterización del FL: las características de los problemas se pueden medir por medio de métricas descriptivas, mientras que las mĺetricas dinámigas están más relacionadas a las características del algoritmo. Esta contribución presenta un procedimiento llamado Modelos de Clasificación de Rendimiento (PCM) el cual crea modelos para predecir el rendimiento exhibido por los Algoritmos Genéticos (AG) en la solución de problemas de optimización en dominios continúos. PCM clasifica el rendimiento en dos clases (fácil o difícil). El conjunto de datos tiene características del FL como variables predictoras, y el rendimiento exhibido por el AG como variable objetivo. Los problemas usados en los experimentos son funciones de optimización de referencia. Un producto de PCM, es un procedimiento para Recomendar Tamaño de Población (RPS): dado un problema de optimización, RPS recomienda la población mínima para obtener un nivel eficiente de rendimiento. Este trabajo puede ser fácilmente extendido para usar otras métricas, se puede aplicar a un conjunto diferente de problemas, o usar otro AE. El desarrollo de modelos de rendimiento para otros AE, puede llevar a la solución de una instancia del problema de selección de algoritmo.

Content

List of Figures

List of Tables

List of Acronyms

ANOVA Analysis of Variance

ASP Algorithm Selection Problem

BBOB Black Box Optimization Benchmark

CEC Congress of Evolutionary Computation

COCO COmparing Continuous Optimizers

CUP Closed Under Permutation

EA Evolutionary Algorithm

FC Fitness Cloud

FDC Fitness Distance Correlation

FL Fitness Landscape

FLA Fitness Landscape Analysis

GA Genetic Algorithms

GECCO Genetic and Evolutionary Computation Conference

GP Genetic Programming

ITTG Instituto Tecnologico de Tuxtla Gutierrez

ITVM Instituto Tecnologico del Valle de Morelia

ITZ Instituto Tecnologico de Zitacuaro

ML Machine Learning

MLP Multilayer Perceptron

NB Naive Bayes

NFL No Free Lunch

NSC Negative Slope Coefficient

PCM Performance Classification Models

RCGA Real-Coded Genetic Algorithms

RF Random Forests

RPS Recommend Population Size

SS Sum of Square

UCTP University Course Timetabling Problem

WEKA Waikato Environment for Knowledge Analysis

List of Symbols

d	Problem dimension.
Ω	Search space on the $\mathbb{R}^d$ domain.
$\mathcal{S}$	Set of points sampled from the search space ($\mathcal{S} \subset \Omega$).
s	A point from $\mathcal{S}$.
f	Function in the continuous domain.
$\mathbf{E}$	Equality constraints.
$\mathbf{I}$	Inequality constraints.
F	Set of functions in the continuous domain.
M_n	Classification model.
ϵ	Precision required in GA.
n	Population size.
n_T	Number of independent executions of GA.
$\mathcal{N}_{\mathcal{S}}(s, \delta)$	The neighborhood function defined on $\mathcal{S}$, where δ is the maximum euclidean distance between s and its neighbors.
$d_E(x, y)$	Euclidean Distance defined in Ω.
δ	Radius of a neighborhood.
γ	Maximum distance between two fitness values considered similar.

L_b	Lower bound of a search space.
U_b	Upper bound of a search space.
f^{max}	The highest fitness value.
$f(x^*)$	The minimum fitness value.
$\mathcal{M}$	Sequence of fitness landscape features.
$\mathcal{P}$	Sequence of performances values.
$\mathcal{D}$	Dataset.

Chapter 1

Introduction

When practitioners try to solve optimization problems, generally, the first step is to determine the type of problem. That is, the encoding, the complexity (e.g. NP problems), the number and type of constraints, etc. The second step is the selection of a solver; when traditional solvers are unable to tackle the optimization problem, the best options is the use of different flavours of Evolutionary Algorithms (EA) [Knjazew12]. In literature, we can find many EA, each algorithm with its specific heuristic and associated parameters. Generally, users adopt the most popular EA using parameters suggested by literature. Clearly, following this procedure is not the best way to solve a particular problem, so it is necessary to develop methods that can select the best algorithm and its associated parameters to solve a particular optimization problem, that is, select the best optimization algorithm based on the difficulty of the optimization problem. One of the most prominent methodologies to tackle this task is the use of Fitness Landscape (FL). Basically, FL is defined as the geometric form depicted by the fitness function (or cost function) of optimization problems.

The FL can be mathematically defined as follows:

$$L_{\mathcal{S}} = (f, \mathcal{S}, \mathcal{N}) \tag{1.1}$$

where f is the optimization problem, $\mathcal{S}$ is the sample search space, and $\mathcal{N}(x)$ is the neighborhood associated to a point x in the search space. The neighborhood uses some type of distance metric, like euclidean distance or genetic distance (number of genetic steps).

Figures 1.1, 1.2, 1.3, and 1.4 show the FL associated to some benchmark functions, and these functions represent minimization problems in two dimensions in the continuous domain.

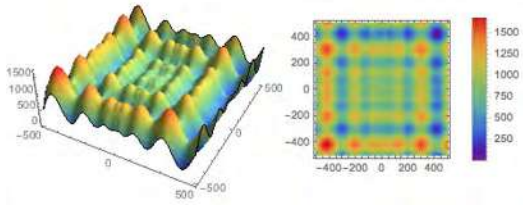

Figure 1.1: Schwefel's function.

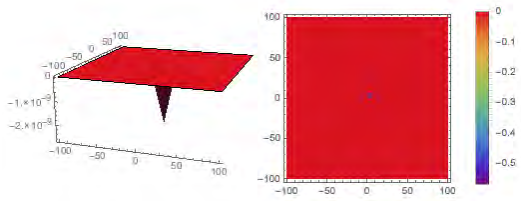

Figure 1.2: Easom Function.

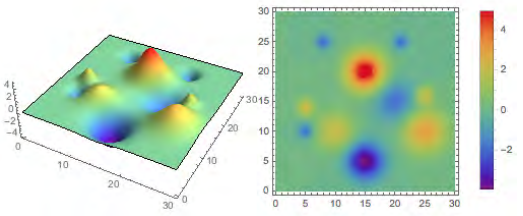

Figure 1.3: Randompeaks function.

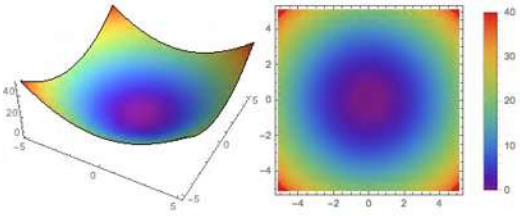

Figure 1.4: Sphere function.

As we can see, Figure 1.1 is a rugged function with multiple local minima, Figure 1.2 presents a smooth landscape with a wide neutral area (equal values of fitness), Figure 1.3 is an example of deceptiveness, due to some attractors (local optima) make difficult the search (deceive the search) to get the global optimum. Finally, Figure 1.4 is a monotonic decreasing function, the function presents a global funnel structure with a single attractor (global optimum).

A characterization of landscapes is performed by Fitness Landscape Analysis (FLA). FLA uses a set of metrics to determine whether a particular problem is easy or hard to solve by a particular EA. At the beginning, the first works related to FLA based their approach on only one FL metric the results were not satisfactory because the metrics only

characterized correctly some problems [Grefenstette92, Naudts00b, Reeves95, Horn94]. Perhaps, the closest metrics related with the difficulty of optimization problems are the metrics that measure the evolvability of EA, that is, in an EA process, the capacity of individuals to improve their fitnesses has a direct relation with the difficulty of the problem.

Recently, the majority of works [Caamaño10, Caamaño13, Malan09, Malan13, Trujillo12b, Trujillo11, Muñoz15] use a set of FL metrics to try to capture the majority of the features of problem and algorithm. These features can give some insight about the difficulty of optimization problems, being possible the creation of learning models to predict performance of the algorithm. The dataset to make the learning models, can take as predictor values, FL measures, and as target values, the performance obtained by the algorithm. The dataset will be composed by the sequences $\mathcal{M}$ and $\mathcal{P}$ as in Equation 1.2.

$$\langle (m_1, p_1), \ldots, (m_k, p_k) \rangle \tag{1.2}$$

where $m \in \mathcal{M}$ is an N-dimensional vector in $\mathbb{R}^N$ containing the FL features, $p \in \mathcal{P}$ in $[0, 1]$ is the performance obtained by the algorithm in the solution of the optimization problems, and k is the number of instances (optimization problems).

This work presents an approach to predict the difficulty of solving optimization problems in the continuous domain in two dimensions using GA. This approach is called *Performance Classification Models (PCM)* [Rodriguez-Maya14a]. The difficulty of optimization problems is approximated to GA performance; the performance values are categorized in two: easy and difficult [Mersmann13, Trujillo12a]. Models use as predictor variables the features: neutrality, ruggedness, basins of attraction, epistasis, fitness distance correlation, and negative slope coefficient. Target variable is the performance obtained by GA. 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 possible values of performance.

PCM can be used to build a recommender-system, that is 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.

1.1. Problem Statement

Given this optimization function f defined as follows:

$$\min f(x),\ f : \mathbb{R}^N \to \mathbb{R}$$
$$U_b \leq x_i \leq L_b,\ i \in [1, N]$$

where x is a candidate solution defined in $\mathbb{R}^N$ within the simple boundary constraints U_b and L_b (upper and lower boundaries, respectively), and problem consist in minimization.

Consider a finite dataset $\mathcal{D} = \langle \mathcal{M}, \mathcal{P} \rangle$ where $\mathcal{M}$ is a sequence of FL features defined in $\mathbb{R}^N$ and $\mathcal{P}$ is a sequences of performance values defined in $\mathbb{R}$. The objective is the creation of a supervised learning model M using the dataset $\mathcal{D}$ to predict the difficulty of f:

$$M : f \to \text{difficulty}$$

where difficulty is the hardness presented by f when is solved by GA. Features refer to the structural properties of f and GA, and performances is defined as the rate of successful GA executions.

1.2. Motivation

A large variety of real-life problems are in the continuous domain, hence the special interest to solve them [Haupt04]. Some examples of real-life problems are: in agriculture, in the automobile industry, and in the renewable energies, among others.

In optimization, especially in combinatorial optimization, the well known No Free Lunch (NFL) theorem [Wolpert97] establishes that there is not an algorithm superior to another, to solve distinct types of optimization problems. Basically, NFL establishes that there is not an algorithm superior to others for the solution of all optimization problems; fortunately, NFL theorems do only apply to classes of problems that are closed under permutation (CUP), so the door is still open for algorithms working on classes of problems that are not (CUP).

It is clear that for a given optimization problem there exist more appropriate algorithms to solve it. The selection of the more appropriate algorithm could be based on a prediction process where the aim is the a priori knowledge of the success or failure (performance) of algorithms before running them; even, this knowledge can also guide us in the determination of the best input parameters. To make the prediction task, an option is the construction of learning models; to construct such models it is necessary the establishment of a dataset containing predictor and target values; the predictor values can be related to the most important features of problems and the target values with the difficulty (performance) of algorithm to solve the problems.

One of the most successful techniques to face the prediction problem is the Fitness Landscape Analysis (FLA). FLA uses a set of Fitness Landscape (FL) metrics to measure different aspects of the landscape depicted by the cost function of the optimization problems. The main questions to be answered with respect to the use of FL metrics are: what are the most accurate metrics?, what are the most desirables metrics, metrics focused in the problem or in the algorithm?, and what is the best method for the usage of a set of metrics?

1.3. Hypothesis

It is possible to create a model that predicts the peformance exhibited by Genetic Algorithms solving continous optimization problems.

1.4. Objectives

The main objective is to develop models capables of predicting the performance of Genetic Algorithms on optimization problems in the continuous domain in two dimensions.

1.4.1. Particular Objectives

To generate the dataset is necessary the usage of fitness landscape features which serve as predictors, and GA performance values for the target values, in both cases, the computation of features and the GA performances are based on benchmark functions in the continuous domain. Once the dataset is conformed, the next step is the selection of the most accurate learning models. A direct output of the learning models is an application to Recommend Population Size (RPS).

1.5. Contributions

The main contributions of this work are the following.

- The development of learning models using as predictor variables, a set of fitness landscape metrics and GA input parameters, and as target variables, GA performance values. In this case, the models obtained an accuracy of 90%.
- The construction of learning models considering only fitness landscape measures as predictive variables, and performance values (averaging the performance on a set of different precisions) as target values. The learning models were based on a set of population sizes with an accuracy around of 72%.
- A direct application of the models is a procedure to recommend population for the GA: for a given optimization problem, the procedure recommends the minimal population size to get an acceptable approximation to the optimal solution.
- A Fitness Landscape Analysis on a real life application was performed. The analysis was performed on three instances of the University Course Timetabling Problem; Based on the analysis, a recommendation of population size for the GA was performed.

1.6. Publications

1.6.1. Journal Papers

- *Models to Classify the Difficulty of Genetic Algorithms to Solve Continuous Optimization Problems*. Noel Rodriguez, Juan J. Flores, Mario Graff, Sébastien Verel. Sent to Soft Computing, Springer, ISSN 1432-7643, Impact factor 1.271.

1.6.2. Conference Papers

- *Predicting the RCGA Performance for the University Course Timetabling Problem*. Noel Rodriguez, Juan J. Flores, Mario Graff. An International Symposium on Intelligent Computing Systems (ISICS), Communications in Computer and Information Science (CCIS) series, Springer. Merida, Mexico, on March 16th - 18th, 2016.

- *Performance Classification of Genetic Algorithms on Continuous Optimization Problems*. Noel Rodriguez, Mario Graff, Juan J. Flores. In Nature-Inspired Computation and Machine Learning, Springer International Publishing, pp. 1-12, Vol. 8857, ISSN 0302-9743, DOI $10.1007/978-3-319-13650-9\_1$. Mexican International Conference on Artificial Intelligence (MICAI) 2014. Tuxtla Gutierrez, Chiapas, Mxico.

- *Solving a Scholar Timetabling Problem Using a Genetic Algorithm Study Case: Instituto Tecnolgico de Zitcuaro*. Noel Rodriguez, Jose Martinez, Juan J. Flores, Mario Graff. In Artificial Intelligence (MICAI), 2014 13th Mexican International Conference on, IEEE, pp. 197-202, DOI 10.1109/MICAI.2014.36. Mexican International Conference on Artificial Intelligence (MICAI) 2014. Tuxtla Gutierrez, Chiapas, Mxico.

1.7. Thesis Outline

Chapter 1 introduces the main concepts and definitions used in this work. Firstly, the problem and motivation are established, followed by the hypothesis and the main objectives, finally the main contributions and related publications are presented.

Chapter 2 establishes the main research works related with two schemas of fitness landscape metrics (descriptive and dynamic) used in this work are also presented.

Chapter 3 defines all the concepts and metrics used in this work. The chapter firstly presents a brief introduction on fitness landscape analysis, then, descriptive and dynamic metrics are defined.

Chapter 4 introduces to optimization methods based on local and global search. Firstly, a definition of optimization tasks is presented, secondly, some methods based on local search are presented, finally, the main method used in this work based on global search (RCGA) is described.

Chapter 5 presents the main contribution of this work, the models to classify the performance of RCGA. The first part states the problem, the second part defines the equations to support the computation, the third part shows the procedure to construct the dataset followed by the Random Forest procedure, and finally is presented the Procedure to Recommend Population size.

Chapter 6 shows the results obtained by the proposal. Firstly, the benchmark problems and parameter setting used are presented. Secondly, the accuracy obtained by the models are described. Thirdly, the algorithm and results of the procedure to recommend population size are analyzed. Finally, a real life application of fitness landscape metrics is presented; the application is based on the characterization and prediction of the University Course Timetabling Problem.

Chapter 7 presents a brief discussion on the results obtained. The main conclusions, and further research are also presented.

Chapter 2

Related Work

This chapter presents some of the most important work for the prediction of the performance exhibited by EA in the solution of optimization problems. Section 2.1 presents a brief introduction, Section 2.2 presents some of the most representatives measures based on problem's characteristics, Section 2.3 reviews some of the most representative measures based on algorithm's evolvability, Section 2.4 presents some representative approaches based on a set of measures to characterize optimization problems, and finally, Section 2.5 presents a brief summary.

2.1. Introduction

The first effort to try to predict the hardness of problems was introduced by Wright in 1932 in the theoretical genetics field [Wright32]. Wright studied evolution through the relationship between the genotype space (organism) and its reproductive success. Wright's approach can be viewed as a metaphor of an optimization process: searching through the search space to find the optimal solution. This metaphor was called Fitness Landscape (FL): where for each point in the search space, corresponds a fitness value, those values depict peaks, valleys, smooth areas, etc. Basically, the FL represents a geometric form of an optimization problem. To perform FL analysis, different features of optimization problems

are considered, e.g., rate of smoothness, and rate of ruggedness, among others. It is clear, for example, that a smooth landscape with a single optimum will be relatively easy to search for many algorithms, while a very rugged landscape, with many local optima, may be more problematic [Horn95, Kauffman91].

A natural way to establish the success or failure of EA in the solution of optimization problems is related to its performance. In this work, we define performance as the rate of success to find the global optimal in a set of trials. In the beginning, researchers tried to predict the performance exhibited by EA using a single FL feature [Grefenstette92], e.g., they tried to predict the performance with the rate of ruggedness of the landscape. In the last decades, researchers have tried to predict the EA's performance with algorithms features [Vanneschi06a], i.e., they tried to relate the performance with the capacity of evolution of individuals in a particular metaheuristic. In both cases, there are no general conclusions, in the majority of approaches, the prediction was acceptable only for some problems. Recently, a large number of works suggest for prediction, the use of a set of metrics to obtain the major features of problems and algorithms. In this sense, thre is necessary the use of metrics to capture the description and dynamic of the search process.

2.2. Approaches Based on Descriptive Problem Features

Deb and Goldberg [Deb93] analized the level of deceptiveness on trap functions using GA; they relate the level of deceptiveness with the hardness of problems. Grefenstette [Grefenstette92] demonstrated that deceptiveness in isolation is not determinant to relate it with the difficulty of optimization problems for GA. Naudts and Kallel [Naudts00b] used epistatis variance and fitness distance correlation as difficulty metrics; they concluded that metrics in isolation are not capable of distinguishing between easy and hard problems. Reeves and Wright [Reeves95] used epistasis as problem difficulty in bit-string problems using GA, their conclusions showed a low relation between problem difficulty and the level of epistasis. Horn and Goldbert [Horn94] performed the first experiments for the relation between the level of multimodality in optimization problems and the difficulty in GA to solve the problems; they concluded that the ruggedness itself, is not determinant of the

difficulty presented by the search space.

In the field of Genetic Programming (GP), there are many models to predict the performance of GP systems on different types of problems. Graff and Poli proposed a performance model [Graff10]; the main idea is that the performance of a GP system for a given problem can be estimated using a set of points from the search space. Later on, Graff et al. proposed another performance model [Graff12], based on the discrete derivative of functions, and, consequently, it has only been tested on symbolic regression problems. These two models have been successfully applied to symbolic regression, boolean induction, and time series forecasting [Graff13], among other types of problems.

In [Huang09], the authors perform an optimization process to find the best allocation in a wireless communication problem (a combinatorial problem); the authors performed a fitness landscape analysis (using ruggedness and fitness distance correlation as FL metrics) to find the appropriate local search, and the appropriate genetic operator within a Memetic Algorithm process; the results confirm the adequacy of fitness landscape analysis to the study of combinatorial problems.

2.3. Approaches Based on EA's Dynamics

EA dynamics (evolvability) refers to the efficiency of the evolutionary search [Vanneschi04a], that is, the level of improvements to find the optimal, in an evolutionary process. One of the first tools to measure the level of improvements in an EA process, was the Fitness Cloud (FC), proposed by Verel et al. [Verel03]; FC measures the level of evolvability depicted by the fitness of individuals against the fitness of its corresponding neighbors in a search process.

Jones proposed a landscape graph-based model [Jones94], basically, the model is a labeled graph (vertex and edges) which is designed to study landscapes from the point of view of the EA evolvability (capacity to walk through the landscape). Jones suggests [Jones95b] that each genetic operator in a GA process describes its own landscape (one operator, one landscape), this implicates that in the evolvability of a search algorithm, it

is necessary the consideration of the genetic operators associated to such algorithm; in the case of GA, the associated genetic operators are: selection, mutation, and crossover.

Jones and Forrest proposed Fitness Distance Correlation (FDC) [Jones95c] as a metric to measure the level of deceptiveness on problems coded as binary strings using GA as solver. Their main proposal relates the difficulty of the search with the deceptiveness of problems: the more deceptive problem, the harder to solve. To compute the deceptiveness, FDC measures the correlation between a set of fitnesses and its corresponding distances to the global optimum; to compute the distances, it is necessary to know a priori the global optimum. FDC was tested in some families of optimization problems with good results, however, there are counter examples about the use of FDC. Müller et al. [Müller11] used FDC as an indicator of the global structure for the problems proposed in the CEC2005 (Congress of Evolutionary Computation) contest, which consisted on the solution of black box optimization problems. They proposed the use of euclidean distances to set the distances in FDC; the results showed that FDC is able to distinguish some global structures and can serve as landscape descriptor.

Related to the concepts of evolvability and Fitness Cloud (FC), Vaneschi et al. [Vanneschi06a, Vanneschi06c] propose the use of Negative Slope Coefficient (NSC) to try to predict the hardness of Genetic Programming problems. NSC tries to determine the evolvability in a search process, such evolvability is related to the hardness of optimization problems. To determine the evolvability, NSC uses slope values plotted on a FC; the hypothesis is summarised as follows: the more negative the slope, the harder the problem [Vanneschi04a]. Vanneschi used the NSC [Vanneschi08] to characterize the difficulty of real life applications (pharmaceutical applications) to select the best Genetic Programming configuration, among a set of configurations. The results showed a reliable GP configuration for all the experiments. Picek et al. [Picek09] proposed an improvement of the Negative Slope Coefficient called New NSC. The proposal is based on the assumption that the original NSC failed to get the difficulty of some discrete optimization problems (unitation functions); the New NSC recalculates the negative slopes through the use of two single points for each segment in the FC. These results showed a strong relation between problem difficulty and

the slope value on the proposed problems.

2.4. Approaches Based on Problem and Algorithm Features

To get a more accurate prediction about the hardness associated to optimization problems using EA, some authors suggest the use of a mixture of features of problems and algorithms. Caamaño et al. [Caamaño10, Caamaño13] used a set FL features (modality and deceptivity) to characterize optimization problems in the continuous domains using three different optimization solvers; the results showed the correlation between separability and multimodality, and the performance obtained by the solvers. Malan et al. [Malan09, Malan13] proposed to characterize problems before trying to predict their performance; based on the problem's characteristics the users can select the best EA to solve the problem.

In some works the authors used FL features to construct learning models to try to predict the hardness associated to optimization problems using a particular optimization solver. Trujillo et al. [Trujillo12b, Trujillo11] made a learning model based on a black box process: from a set of Genetic Programming problems, their FL features were passed as predictor variables and the performance obtained by the solver as the target value. In their study, they used dynamic and static FL features, and concluded that the static features were more correlated with the performance. Mario Muñoz et al. [Muñoz15] proposed a classification model based on a data driven method, Information Content of Fitness Sequences (ICoFiS). They efficiently categorized real optimization problems. Malan et al. [Malan14] created a learning model for the classification of the performance of 7 different PSO variants in two classes. The dataset consists of 24 benchmark functions with different dimensionality, having a total of 116 problem instances. As predictive attributes, the model uses 11 fitness landscape features (including the dimensionality) and four kinds of performance measures. The results showed accurate models with an accuracy above 90%.

Recently many works have been emerging due to the popularity of the Genetic and Evolutionary Computation Conferenc (GECCO) workshop Black Box Optimization Benchmark (BBOB) [Auger12] and more recently at the IEEE Congress on Evolutionary

Computation (CEC'2015), where the participants test the performance of their heuristics COmparing Continuous Optimisers (COCO) [Hansen10]. COCO is a real-parameter optimization platform where the users can prove systematically diverse optimization solvers on a set of real-optimization problems (benchmark functions). Olaf Mersman et al. [Mersmann11] proposed the use of a reduced number of fitness landscape features (low-level features) for the classification of the BBOB, they classified efficiently the problems into 6 categories. More recently Bischl et al. [Bischl12] suggested the use of low-level features proposed by Mersman et al. [Mersmann11] to characterize the fitness landscape of BBOB functions. For the characterization they used the Exploratory Landscape Analysis (ELA) where the characterization is performed prior to optimize. Another approach is based on cell mapping techniques (global behavior of non linear dynamical system) [Kerschke14], the experiments showed interesting dynamic characteristics of fitness landscapes. In [Asmus14] the authors propose the use of Formal Concept Analysis, and the set of functions contained in BBOB for the construction of a recommended system for black optimization. They use fitness distance correlation [Jones95c] and ICoFiS [Muñoz15] as fitness landscape metrics. Their goal is the recommendation of the most suitable algorithm for the solution of a given problem.

2.5. Summary

This chapter introduced some of the most relevant work on the characterization of optimization problems, and the prediction of performance of EA. Currently, there is not a final conclusion about the best set of features for the characterization and prediction of performance; the most accurate approaches are based on a set of features of problems and algorithms. The majority of approaches try to determine the most representative fitness landscape features for characterization and prediction, generally on a small set of predefined functions, while in this work we perform the characterization using a large set of benchmark functions. Additionally, we employ the resulting models for the construction of a system to recommend population size.

Chapter 3

Real-Coded Genetic Algorithms for Optimization

Optimization is the process of finding the best element, between a set of alternatives, that maximizes a profit criterion. Solvers are algorithms that perform the optimization process. Some of the most popular and efficient solvers are based on populations, particularly on EA; hence the interest for the study of its behavior. In this chapter, Section 3.1 introduces to the optimization concept, Section 3.2 provides a brief introduction to EA, particularly to GA, finally, a brief summary is presented in Section 3.3.

3.1. Introduction

Optimization is the process of finding the best value or set of values that maximize or minimize a given criteria, generally, specified as a function (optimization problem). The first step for the establishment of optimization problem is to model it (artificial or real). Modeling is the process of identifying the objective, variables, and constraints for a given problem; once a model has been formulated, an optimization algorithm can be used to find its solution [Nocedal06].

Formally, for a given function f, subject to the constraints c_i, optimization is the

process to find the variable x that maximizes or minimizes the function (see Equation 3.1).

$$\begin{aligned} &\arg\min_{x \in \mathbb{R}^N} f(x) \\ &\text{subject to} \\ &c_i = 0, \ i \in \mathbf{E} \\ &c_i \geq 0, \ i \in \mathbf{I} \end{aligned} \tag{3.1}$$

where x is defined in the $\mathbb{R}^N$ domain, $\mathbf{E}$ and $\mathbf{I}$ are set of indices for equality and inequality constraints, respectively.

Generally, optimization problems are classified within discrete, continuous and hybrid search spaces. The variables in the discrete optimization problems are integers from a finite set (often very large), while for continuous optimization problems, the variables are uncountably infinite, e.g. the set of real numbers [Nocedal06], and hybrid search spaces which combining both types of variables. In literature we can find optimization methods based on local search and others based on global search; in the first case the solver finds only local solutions without taking into consideration global solutions, in the second case the solver considers the global properties of the search within the predefined bounds. The following section describes one of the most popular global search methods: Evolutionary Algorithms.

3.2. Evolutionary Algorithms

Evolutionary Algorithms (EA), are population-based methods; to perform the search process, these methods use a set of individuals or candidate solutions, those individuals evolve through time until an approximation to the optimum is reached [Blum03]. EA use mechanisms inspired by biologic evolution such as: reproduction, mutation, recombination, and selection. Generally, EA perform well approximating solutions to different types of problems, because they do not make any assumption about the underlying fitness lands-

cape. At the beginning of the computation, the population are randomly initialized, the objective function is evaluated for these individuals, then, the first generation is produced; if the optimization goal is not reached, a new generation is produced.

EA include a group of several global optimization methods inspired by the Darwinian principle of nature's capability to evolve through time to adapt to the environment [Boussaïd13]. Some of EA's most representative methods are: Genetic Algorithms, Evolutionary Strategies, Evolutionary Programming, Genetic Programming, Differential Evolution, and Cultural Algorthms.

3.2.1. Real-Coded Genetic Algorithms

Genetic Algorithms (GA) were invented by John Holland in the 1960s at the University of Michigan. Holland's original goal was not to design algorithms to solve specific problems, but rather to formally study the phenomenon of adaptation as it occurs in nature and develop ways in which the mechanisms of natural adaptation might be imported into computer systems. Holland's GA is a method for moving from one population of individuals represented by chromosomes -binary strings- to a new population by using a kind of natural selection together with the genetics-inspired operators of crossover, mutation, and selection. Each chromosome consists of genes (i.e., bits), each gene is an instance of a particular allele (i.e., 0 or 1). The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones. Crossover exchanges subparts of two chromosomes, roughly mimicking biological recombination between two single-chromosome (haploid) organisms. Mutation randomly changes the allele values of some locations in the chromosome [Mitchell96].

GA were originally developed to operate on bit-strings, if a GA user wants to solve a continuous optimization problem, a codification-decodification process is necessary. Due to its discrete representation, GA have difficulties when dealing with continuous search spaces with large dimensions and high precision [Herrera98]. Generally, real-life optimization problems are coded using continuos domains; for these types of problems, several optimization algorithms have been developed. Some examples of evolutionary algorithms using real

encoding are: ant colony optimization, artificial bee colony algorithm, evolution strategies, differential evolution, and particle swarm optimization, among others. GA with a continuous encoding is known as the Real-Coded Genetic Algorithms (RCGA), where its variables are a direct representation of the continuous search space.

Literature reports that, for some problems, the real-coded representation and associated techniques outperform the conventional binary representation [Yoon12]. RCGAs have shown their ability to solve a wide variety of real-world problems, they have been applied to parameter estimation, neural networks, aerospace design, biotechnology, economics, and constrained parameter optimization problems [Ortiz-Boyer07]. The performance of GA depends on the operators selection, mutation (type and rate), and crossover (type and rate). Population size plays a crucial role in the performance of GA: a small population finds good solutions, but it often gets stuck on local optima [Spears91].

Algorithm

The general operation of GA starts with a random population (chromosomes) selected from the search space, the population advances toward better individuals by applying genetic operators (selection, mutation, and crossover), then the population is replaced by the new one; this iterative process is called generation [Herrera98]. The GA general process consists of three basic operations [Herrera98]: evaluation of individual fitness, formation of a gene pool (intermediate population) through a select mechanism, and recombination through crossover and mutation operators. Finally, the best solution is returned. Algorithm 1 shows the general GA process.

Algorithm 1 Genetic Algorithm.

```
GA()
1  t ← 0
2  initialize  P(t)
3  evaluate  P(t)
4  WHILE not termination
5    P'(t) ← recombine  P(t)  //crossover and mutation
6    evaluate  P'(t)
7    P(t + 1) ← select  P'(t)
8    t ← t + 1
9  return  best_solution
```

Representation is a key in the GA process, related with the level of expresiveness [Herrera98]. Some examples of GA representation are: Vectors of floating point numbers, vectors of integer numbers, and ordered list, among others.

3.2.2. Objective Function

In GA the objective function or cost function is the objective of problem; in minimization problems is the model where we want to find its minimum value [Chipperfield94]:

$$f : \mathbb{R}^N \rightarrow \mathbb{R}$$

where f is the objective function, $\mathbb{R}^N$ is the N-dimensional input parameter on the continuous space, and $\mathbb{R}$ is the output value on the continuous space.

Genetic Operators

The genetic operators guide the search towards solutions to a given optimization problem. Generally, GA are related with three genetic operators: selection, mutation, and crossover.

- Selection determines which individuals are chosen for recombination and how many

offspring each selected individual produces. There are many types of selection methods [Mitchell96]: Fitness-Proportionate Selection, Sigma Scaling, Elitism, among others.

- Crossover is a method for sharing information between chromosomes, through the mating of two parent chromosomes to form two offsprings [Herrera98]. Different types of crossover operator have beed developed for RCGA [Ortiz-Boyer07, Chipperfield94, Magalhaes-Mendes13, Kaya11]: Multi-point crossover, Flat crossover, Arithmetical crossover.

- Mutation alters some components (genes) of a selected chromosome to increase the exploration capacities of population [Herrera98]. There are two types of mutations: bounded and unbounded; for bounded, the most used are creep mutation and single-variable mutation, and for the unbounded, the Gaussian mutation is the most used [Yoon12].

3.3. Summary

This chapter introduces to optimization and Evolutionary Algorithms. In a special manner, the Real-Coded Genetic Algorithm (RCGA) was described, since RCGA is the solver to be used in this work. To make an evolutionary algorithm efficient, it is necessary to characterize the problem to be optimized. There are some guides to characterizate the optimization problems; the following chapters establish some of the basis for this porpuse.

Chapter 4

Fitness Landscape Analysis

One of the first steps to predict the success or failure of EA is the characterization of the optimization problem to be solved. One of the most successful techniques to characterize an optimization problem is by performing a Fitness Landscape Analysis. This chapter presents a general description of FLA, and the definition of the metrics used in this work. Section 4.1 introduces to FLA, Section 4.2 defines the descriptive FL metrics, and Section 4.3 defines the dynamic FL metrics.

4.1. Introduction

In optimization and particularly in the field of EA, one of the most important questions to be answered is, what features or set of features make an optimization problem difficult to solve? In a natural way, the user generally makes assumptions about the hardness of optimization problems; problems with a funnel structure (as the sphere function) generally are easy to solve and problems with many local optimal (as Schwefel's function) are difficult to solve. To predict the complexity presented by the optimization problems, it is necessary the understanding of global and local structures presented in landscapes; this understanding is commonly known as characterization. If we are able to characterize most features of optimization problems, we can construct a learning model to predict the success

or failure of the solvers (e.g. GA).

Currently, there is not a successful technique in the EA field, that ensure an accurate estimation of difficulty for the majority of optimization problems [Malan09]. Fitness Landscape Analysis (FLA) groups a set of FL metrics for the characterization of optimization problems. Those metrics are capable of measuring different features of problem and solver, for the purpose to provide insights about the hardness presented by the solver in the solution of optimization problems [Malan09].

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 they are solved by EA [Caamaño10, Jones95a]. It is necessary the usage of a set of FL features, that ensure a more accurate prediction. 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, this work proposes the use of two types of FL metrics: descriptive and dynamic [Malan09, Merkurjeva11, Reidys02]. Descriptive metrics focus on the problem's features while dynamic metrics focus mainly on the algorithm.

4.2. Descriptive metrics

The descriptive metrics used in this paper are: *Ruggedness*, *Neutrality*, *Basins of attraction*, and *Epistasis*. The rest of the sub-section describes those metrics.

4.2.1. Neutrality

Neutrality was introduced in biological evolution theory by Kimura [Kimura83]. In the field of EA, neutral regions are areas of the FL that have similar fitness values [Galván-López06], i.e. similar fitness values within a neighborhood. Neutrality is the rate of neutral areas in $\mathcal{S}$; in Equation 4.1 we compute an estimation of neutrality based on the sample $\mathcal{S}$. To compute neutrality, we estimate the rates of equal fitness (neutral regions)

contained in the neighborhoods of $\mathcal{S}$.

$$\text{neutrality}_{\mathcal{S}}(\delta,\gamma) = \frac{\sum_{s\in\mathcal{S}} \frac{|\mathcal{NN}_{\mathcal{S}}(s,\delta,\gamma)|}{|\mathcal{N}_{\mathcal{S}}(s,\delta)|}}{|\mathcal{S}|} \tag{4.1}$$

where $\mathcal{S}$ is a set of points from the search space, δ is the maximum distance between neighbors, 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 4.2), and $\mathcal{N}_{\mathcal{S}}(\cdot)$ is the neighborhood function defined in Equation 4.3.

$$\mathcal{NN}_{\mathcal{S}}(s,\delta,\gamma) = \{\forall s' \in \mathcal{S} | s \neq s' \wedge d_E(s,s') \leq \delta \wedge d_E(f(s),f(s')) \leq \gamma\} \tag{4.2}$$

$$\mathcal{N}_{\mathcal{S}}(s,\delta) = \{s' \in \mathcal{S} | s \neq s' \wedge d_E(s,s') \leq \delta\} \tag{4.3}$$

High rates of neutrality, are not desirable in an FL to produce a suitable evolutive environment [Smith02], i.e. neutrality can affect the distribution of local optima and as a consequence the success of searching [Malan13].

4.2.2. Ruggedness

Ruggedness is a measure related to the number of peaks surrounded by valleys in a FL; a problem has a high degree of ruggedness when the fitness function in the search space has a high rate of changes [Vassilev99, Lobo04, Pitzer12]. In a rugged FL, the individuals of many EA can get trapped in local optima as a consequence of premature convergence [Malan13]; generally, the more rugged a function is, the harder to optimize it [Weise09]. There are many techniques to measure the level of ruggedness [Weinberger90, Lipsich91], however the information retrieved is be considered as basic and not reflect problem difficulty [Mattfeld99]. Vassilev et al. [Vassilev00] propose the use of information theoretic technique

for analyzing the ruggedness of discrete fitness landscapes based on entropy, Malan et al. take the Vassilev's proposal to be used in continuous domain [Malan09]; this work is based on the adaptation of Vassilev's work proposed by Malan on the continuous domain.

Entropy measures ruggedness by means of three-point paths; a three-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 [Malan09]. 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 the 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(4.4).

$$s_i = \Psi_{\phi_t}(i, \gamma) = \begin{cases} \bar{1}, & \text{if } \phi_i - \phi_{i-1} < -\gamma \\ 0, & \text{if } |\phi_i - \phi_{i-1}| \leq \gamma \\ 1, & \text{if } \phi_i - \phi_{i-1} > \gamma \end{cases} \tag{4.4}$$

Parameter γ is a real number that determines the accuracy of the calculation for $S(\gamma)$. Equation (4.5) estimates the ruggedness rate through the entropic measure $H(\gamma)$ exhibited by the sequence S.

$$H(\gamma) = -\sum_{p \neq q} P_{[pq]} log_6 P_{[pq]} \tag{4.5}$$

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 (rugged shapes). $P_{[pq]}$ is defined according to Equation 4.6.

$$P_{[pq]} = \frac{n_{[pq]}}{n} \tag{4.6}$$

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. $H(\gamma) \in [0, 1]$ is a rate of the variety of shapes present in the Fitness Landscape. The higher the value of $H(\gamma)$, the wider the variety of rugged shapes in S [Malan09].

4.2.3. Basins of Attraction

Basins of attraction are areas in the search space that lead to a local optimum [Pitzer10]. 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 [Xin09]. In this work, it is approximated the set of basins of attractions by the proportion of local optima found in $\mathcal{S}$; Equation 4.7 computes the proportion of local optima found in $\mathcal{S}$, based on the neighborhood $\mathcal{N}_{\mathcal{S}}$.

$$\mathcal{B}_{\mathcal{S}}(\delta) = \frac{|\{s \in \mathcal{S} | \mathcal{H}(s, \delta)\}|}{|\mathcal{S}|} \tag{4.7}$$

δ is the maximum distance between neighbors, $\mathcal{H}(\cdot)$ is a hill-climber algorithm that calculates the number of local optima (attractor) for each point in $\mathcal{S}$ (see Equation 4.8).

$$\mathcal{H}(s, \delta) = \{s' \in \mathcal{N}_{\mathcal{S}}(s, \delta) | f'^{*} \leq f(s')\} \tag{4.8}$$

where $\mathcal{N}_{\mathcal{S}}(\cdot)$ is the neighborhood function defined in Equation 4.3, and f'^{*} is the minimum fitness value found in the neighborhood. A procedure for determining the attractors is mentioned in [Ochoa08].

4.2.4. Epistasis

Epistasis was introduced in GA by Davidor [Davidor90] 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. [Chan03] 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 4.9 measures the level of epistasis (contribution) of the variables x and y in a cost function f.

$$SS_{xy} = \frac{1}{|\mathcal{S}|}\sum_{x\in\mathcal{S}}\sum_{y\in\mathcal{S}} f(x,y) - \frac{1}{|\mathcal{S}|^2}(\sum_{x\in\mathcal{S}}\sum_{y\in\mathcal{S}} f(x,y))^2 - SS_x - SS_y \tag{4.9}$$

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 4.10 and 4.11 compute the level of contribution for the variables x and y, respectively.

$$SS_x = \frac{1}{|\mathcal{S}|}\sum_{x\in\mathcal{S}}(\sum_{y\in\mathcal{S}} f(x,y))^2 - \frac{1}{|\mathcal{S}|^2}(\sum_{x\in\mathcal{S}}\sum_{y\in\mathcal{S}} f(x,y))^2 \tag{4.10}$$

$$SS_y = \frac{1}{|\mathcal{S}|}\sum_{y\in\mathcal{S}}(\sum_{x\in\mathcal{S}} f(x,y))^2 - \frac{1}{|\mathcal{S}|^2}(\sum_{x\in\mathcal{S}}\sum_{y\in\mathcal{S}} f(x,y))^2 \tag{4.11}$$

4.3. Dynamic metrics

Dynamic metrics try to capture the difficulty of optimization problems from the point of view of the algorithm, that is, the level of algorithm's evolvability. Wagner and Al-

tenberg [Wagner96] define evolvability as the genome's ability to produce adaptive variants[1] on genetic systems; the adaptive changes depends critically on the genotype-phenotype map. 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). This work uses two of the most successful metrics: Fitness Distance Correlation, and Negative Slope Coefficient. The following subsections describe these metrics.

4.3.1. Fitness Distance Correlation

Fitness Distance Correlation (FDC), was developed by Jones and Forrest [Jones95c]; it 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 [Chen08]. The main advantage of FDC is that it has been proved, as a suitable indicator of problem difficulty in GA and Genetic Programming [Vanneschi02, Vanneschi06a, Vanneschi05, Pitzer12]. Its main disadvantage is that the optimal solutions must be known a priori, which is unrealistic in most applications [Vanneschi04b, Altenberg97, Naudts00a, Vanneschi05]. Nonetheless, this work uses this metric because the global optimum is known 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 4.12.

$$fdc = \frac{C_{\Phi D}}{\sigma_\Phi \sigma_D} \tag{4.12}$$

[1] Variation is the present differences among the individuals in a population [Wagner96]

$$C_{\Phi D} = \frac{1}{n}\sum_{i=1}^{n}(\phi_i - \bar{\phi})(d_i - \bar{d}) \tag{4.13}$$

where σ_Φ and σ_D are standard deviations, $\bar{\phi}$ and $\bar{d}$ are means of Φ and D, respectively, and the covariance of Φ and D is defined by Equation 4.13.

Jones proposes the use of the Hamming distance as the distance associated with individuals and the optimal solution in a bit-string GA context. In a more general evolutionary computation context, they suggest as measure of distance the use of genetic operators that count the number of steps of indiviuals to reach the global optimum [Jones95c].

4.3.2. Negative Slope Coefficient

NSC was developed by Vanneschi et al. [Vanneschi06a] to capture the evolvability of EA; evolvability is the capacity of genetic operators to improve the fitness quality of individuals [Pitzer12]. 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) [Verel07]. 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 [Pitzer12, Vanneschi06a, Vanneschi06b], and that NSC does not converge for large samples [Vanneschi09].

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 (4.14).

$$FC = \{(f(x_j), f(v_k^j)), \forall j \in [1, n], \forall k \in [1, m_j]\} \tag{4.14}$$

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 4.15.

$$nsc = \sum_{i=1}^{k-1} min(0, P_i) \tag{4.15}$$

Vanneschi et al. [Vanneschi06a] proposed the following hypothesis with respect to *nsc*: negatives values correspond to difficult problems, and values equal to 0 correspond to easy problems.

4.4. Fitness Landscape Computation

To illustrate the metrics described in this section, we performed a Fitness Landscape Analysis on three benchmark functions (complete definition in Appendix A.1); the following figures represent functions in two dimensions: Rastrigin, sphere and Easom.

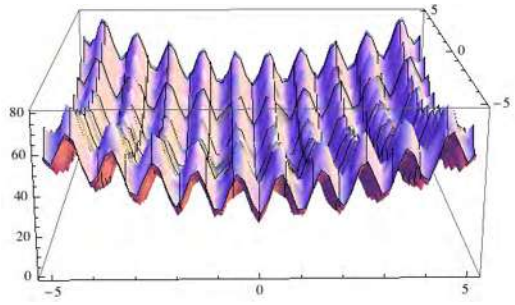

Figure 4.1: Rastrigin function.

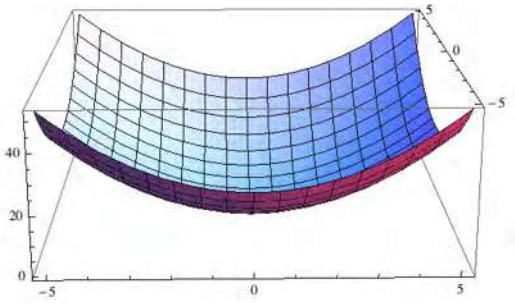

Figure 4.2: Sphere Function.

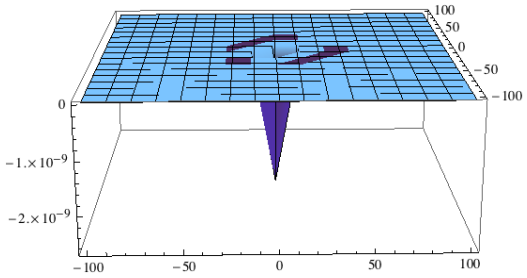

Figure 4.3: Easom Function.

Figure 4.1 presents a rugged landscape, Figure 4.2 presents a single funnel structure, and Figure 4.3 presents a smoothness landscape. To compute the FLA on continuous domains, it is necessary sample the search space for each function. E.g. the Rastrigin function is defined within the boundaries $[-5.12, 5.12]$, then we select 1000 points using random uniform distribution and we calculate its fitness. The next step is the calibration of input parameters for the FL metrics, based in our experiments, the following Table 4.1 shows the parameters and its corresponding values for each metric.

Table 4.1: Input parameters for the Fitness Landscape metrics.

FL metric	**Parameter-value**
Neutrality	$\delta = d_E(L_b, U_b) \times 0.1$, $\gamma = \lvert f^{max} - f^{min} \rvert \times 0.001$
Ruggedness	$\gamma = \lvert f^{max} - f^{min} \rvert \times 0.001$
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, FC=10 bins

Neutrality, Ruggedness and Neutrality use δ and γ parameters: the δ parameter refers to the euclidean distance between individuals to be considered as neighbors within the boundaries of the search space. In this case we set the distance to 10 % of lower and upper bounds, the γ parameter refers to the level of sensibility of fitness values to be considered as equals, we considered $\frac{1}{1000}$ of the maximum distance between the lowest and maximum fitness values. f^{min} and f^{max} are the minimum and maximum fitness values, respectively. Epistasis needs only the fitness function to compute the level of interaction between variables. FDC uses a set of distances which are related to the number of genetic steps (genetic operators) to reach the global optimum; to calculate such distances, we multiply the euclidean distance of individuals and the global optimum by the euclidean distance of the fitness of individuals and global optimum. NSC uses a set of individuals' fitness values and the fitness values of their neighbors (the neighbors are calculated using one step of generation of GA). This set of fitnesses of individuals and their corresponding neighbors is called Fitness Cloud (FC). The FC is divided into bins (10 bins) which serve as centroids to calculate a set of slopes (in this case 9 slopes), and finally calculate the NSC.

Once the input parameters are calculated, the next step is the computation of the metric, e.g., to compute NSC it is necessary the determination of 10 neighbors for each individual (point). The neighbors are obtained applying genetic operators (mutation and crossover) on each individual. Then we organize the fitness of individuals against fitness of its neighbors in a two-dimensional space which is called Fitness Cloud. Next it is divided

the FC into bins: the boundaries of x-axis is divided into 10 bins (equally spaced), then, for each bin is calculated its corresponding centroid (the medium point in x and y axis). Finally, the centroids serve as points to calculate sum of the slopes (NSC) between the bins, staring from the largest to the smallest fitness in the x-axis.

The following Table 4.2 summarizes the computation of the features.

Table 4.2: FLA on the UCTP instances.

	Ruggedness		**Neutrality**		**Epistasis**		**B. of attrac.**		**NSC**		**FDC**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Rastrigin	0.57	0.02	0.02	0.0	0.0	0.0	0.32	0.09	-0.73	0.57	0.90	0.0
Sphere	0.52	0.02	0.03	0.0	0.0	0.0	0.17	0.028	0.0	0.0	0.98	0.0
Easom	0.0	0.0	0.97	0.0	0.98	0.0	0.76	0.01	-36.43	4.20	0.02	0.06

The experiments were repeated 100 times using 1000 points picked randomly from uniform distribution over the search space for each function.

4.5. Summary

This chapter described the metrics used for the characterization of fitness landscape. In this work we proposes the categorization of fitness landscape metrics into two types: descriptives and dynamics. Descriptive metrics measure the problems' features, while dynamic metrics measures the behaviour of the searching process in GA. To visualize in more detail the Fitness Landscape Analysis, a characterization of three benchmark functions were performed using the proposed metrics, these metrics are the basis to support this work.

Chapter 5

Performance Classification Models

Predicting the performance of EA is a difficult task, literature presents a vast number of works related to the prediction of performance of EA on optimization problems. Nonetheless, there is not a final conclusion about this topic. This chapter states the main contribution of this work, the models to predict the performance of GA on continuous optimization problems in two dimensions. Section 5.1 introduces to the proposed model, sub-Section 5.1.1 presents the problem statement, sub-Section 5.1.3 describes the procedure for the generation of the dataset used by the models, sub-Section 5.1.4 shows the generated models, and Section 5.2 describes the Procedure to Recommend Population Size.

5.1. Introduction

Predicting the performance of EA when solving optimization problems, is a difficult problem. The predictor must consider different aspects that affect the search, e.g., which are the most desirables problem features to perform a good search?, or what are the most successful algorithm parameters for a given problem?.

This chapter presents a procedure called *Performance Classification Models* (*PCM*), which creates models to classify the performance obtained by GAs on continuous optimization problems in two dimensions. The models are based on supervised methods, particularly

Random Forests[1]. The models use FL features as predictor variables; those features are derived from the objective function of optimization problems. We use FL metrics described in the previous section: neutrality, ruggedness, basins of attraction, epistasis, fitness distance correlation and negative slope coefficient. These metrics capture both, problems' and algorithms' 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, difficulty is approximated by the performance obtained by the GA to solve it. In this work, performance is defined as the success rate for reaching the global optimum given an error margin; to perform the experiments, the performance is discretized and classified into two categories: easy, and difficult. Then, the target values for the learning models are the discretized performances. The following paragraphs explain in more detail the proposed procedure.

5.1.1. Problem statement

This work proposes a solution to the following problem: given an optimization problem involving function f, derive models capable of predicting the difficulty encountered by GA when solving it. We propose the use of several supervised learning models, the models are related to population size and precision (error margin computation). The problem can be solved by means of two approaches: using regression models or classification models. In the first approach the idea is to find a regression model capable to predict the performance of a given optimization function as in Equation 5.1.

$$M_n : F \rightarrow p \tag{5.1}$$

where M_n is a regression model related to a certain population size n, F are optimization functions on the continuous domain, and p is the performance obtained by the GA in the

[1]Several learning methods were tested: Naïve Bayes, Multilayer Perceptron, Decision Trees, and Random Forests, being Random Forests the most accurate.

solution of F. The second approach is based on classification models (PCM); formally, the models M_n are supervised models that map from a set of problems F to a set of difficulty levels. See Equation (5.2).

$$M_n : F \rightarrow \{easy,\ difficult\} \tag{5.2}$$

where M_n refers to learning models related to a certain population size n, F is a set of optimization functions on the continuous domain, and [easy and difficult] are the difficulty of the GA in the solution of F.

The two approaches make use the concept of performance, in this work performance is defined as the rate of successful GA trials, where the global optimum was found: the number of trials, n_T, is set to 100 and each trial contain a maximum of 1000 generations for the GA. Due to performance is related with the GA operation, we related the performance P with two GA input parameters: 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 population sizes $\mathcal{N} = \{50i\}, i \in [1, 10]$ and precisions $\mathcal{E} = \{10^{-i}\}, i \in [5, 10]$. Perfomance is defined by Equation 5.3.

$$P_{n,\epsilon}(f) = \frac{|\{x | x = GA_{n,\epsilon}(f) \text{ and } |f(x) - f(x^*)| \leq \epsilon\}|}{n_T} \tag{5.3}$$

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, that is, performance is the rate of successful trials to find the global optimal according to certain population size and precision. To discretizes the performance values, we make use the Function $DI(\cdot)$ defined in Equation (5.4). Basically $DI(\cdot)$ cathegorize: easy, to problems where its performance is less than or equal to 0.5, and difficult, to problems where its performance is greater than 0.5.

$$DI(p) = \begin{cases} \text{difficult}, & \text{if } 0 \leq p \leq \frac{1}{2} \\ \text{easy}, & \text{if } \frac{1}{2} < p \leq 1.0 \end{cases} \tag{5.4}$$

To construct the learning models it is necessary the computation of the FL features on F; the function Metrics of Difficulty (MoD) defined in Equation 5.5 computes the FL features for a given problem specified by its objective function f.

$$MoD : F \rightarrow \mathbb{R}^6 \tag{5.5}$$

where F are optimization problems specified as functions on the continuous domain, and $\mathbb{R}^6$ is a six-dimensional vector in the continuous domain, the vector contains the corresponding computation of FL features (ruggedness, neutrality, epistasis, basins of attraction, negative slope coefficient and fitness distance correlation). FL metrics are defined in Chapter 4.

5.1.2. Average Performance

GA may or may not be successful in determining the optimum of a given problem; since both the domain and range of the set of problems addressed in this work are real-valued, it is very unlikely that any metaheuristic used for this optimization process to get the exact optimum. Given that, we need to relax the optimization problem and obtain a quasi-optimal value, to a given precision. In this scenario it is necessary take into consideration different levels of precision, according to the optimization problem, to measure the search ability in a more integral form, i.e. for a given problem where its optimal global value is $f(x^*) = 0$ we can set the precision to 10^{-1} or 10^{-2}, etc. and we measure the algorithm capacities according to different levels of requirement.

Another important input parameter for GA is the population size, it is clear that varying this value we can get different outputs: generally, to more population size we get better results, that is, varying the population size we get different approximation to the global optimum. For instance for a given optimization problem with a single-funnel global

optimum (easy problem) varying the population size perhaps is not relevant than in optimization problems with multiple global optima (hard problem). In this work we set the crossover and mutation rates to 70 % and 30 % respectively, depending the GA search ability to only two parameters: precision and population size.

In previous work [Rodriguez-Maya14a] we performed different performance values considering different population size and precision values: for a given optimization function f we computed the performance using different population sizes (PS), $PS = \{50, 100, ..., 500\}$ and different precision values (PV), $PV = \{10^{-i}\}, i \in [1, 10]$ for the GA. Then, the performances obtained were discretized into three categories and we called to these values Relative Difficulty (RD). $RD(\cdot)$ discretized a performance value into three categories as in Equation 5.6.

$$RD(p) = \begin{cases} \text{difficult}, & \text{if } 0 \leq p \leq \frac{1}{3} \\ \text{medium}, & \text{if } \frac{1}{3} < p \leq \frac{2}{3} \\ \text{easy}, & \text{if } \frac{2}{3} < p \leq 1.0 \end{cases} \tag{5.6}$$

To construct the learning model, the training used as predictor variables the six FL features described in section 4 and the combination of PS and PV, and as objective values, the discretized performance values, those values were performed on 110 benchmark optimization problems containing a total of $11,000$ rows ($110 \times 10 \times 10$ functions, PS and PV, respectively). The experiments were performed using different machine learning techniques implemented in WEKA [Hall09], using 10-fold cross-validation to measure its accuracy. The results showed that Random Forest obtained the best accuracy with 96 % of correct classification.

Despite the accuracy of models created from the data described above, an ideal data set must consider only FL features as predictors and its corresponding performance value as objective. As we have discussed the performance associated to optimization problems is relative to the input parameters to the algorithm, in this case, the input parameters to

the GA. Then, it is necessary a mechanism to extract the most valuable information related with the performance: average performance. 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 (precision) affects the final performance, then an indicator of performance considering those variations is the average performance. For instance, the following Figure 5.1 shows the computation of average performance for different population sizes on 110 benchmark optimisation problems.

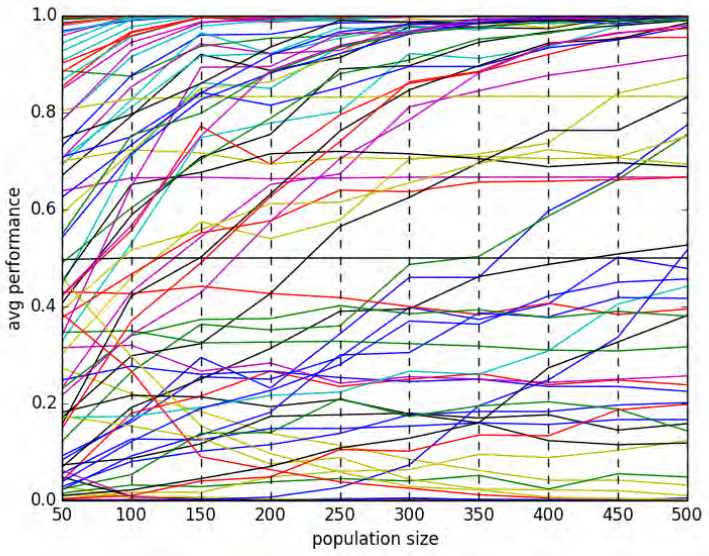

Figure 5.1: Average performance for ech $f \in F$ considering different population sizes.

where the x axis are the population sizes, and, the y axis the normalized average 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.

5.1.3. 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 5.5), and $\mathcal{P}$ are the performance values of GA when solving functions in F (see Equation 5.4). The idea is to make as many models as population sizes; i.e., 10 learning models. To make 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 performances*. The average performance is computed from the set of performances described by $GA_{n,\epsilon}$ for $\epsilon \in \mathcal{E}$. Algorithm 2, *Dataset*, generates the training set to produce a model M_n.

Algorithm 2

```
DATASET(n)
 1  F = {110 optimization problems };
 2  E = {1 × 10^-5, 1 × 10^-6, ..., 1 × 10^-10};
 3  M = ⟨⟩;
 4  P = ⟨⟩;
 5  FOR ALL f ∈ F
 6    sum = 0;
 7    FOR ALL ε ∈ E
 8      sum = sum + P_{n,ε}(f);
 9    avg = sum/|E|;
10    M = append (M, MoD(f));
11    P = append (P, avg) or append (P, DI(avg)); //regression or classification
12     return ⟨M, P⟩;
```

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 5-9). For each $f \in F$, a 6-dimensional array representing its FL features is mapped through the MoD function (see Function 5.5) and stored in $\mathcal{M}$ (line 10), and

its corresponding average performance is established (in the case of classification, the performance is discretized through DI function (see Equation 5.4)) and stored in $\mathcal{P}$ (line 11), when, all the functions are considered to form the dataset, the dataset $\langle \mathcal{M}, \mathcal{P} \rangle$ is returned by the procedure (line 12).

5.1.4. Model

The models (M_n) generated by PCM are based on Random Forest (RF) technique; basically RF is an ensemble of random decision trees generated from the input dataset, the final output is a combination of votes between the trees. The following Algorithm 3 shows the basic operation of RF according to [Breiman01].

Algorithm 3 Random Forest method.

```
RF_Ensemble(D)
1  FOR b = 1  to  B
2     select a bootstrap sample  d  of size  m  from  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     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). In the case of regression, the predicted value at a node is the average response variable for all observations in the node, and for classification, the predicted class is the most common class in the node (majority vote).

5.2. Procedure to Recommend Population Size

One of the major problems when EA users face a new optimization problem is the establishment of the input parameters, generally, they use the most used in literature or the most successful on their projects. Some of the most common input parameters in GA are: crossover rate, mutation rate, and population size [Stanhope98, Vasconcelos01]. Particularly, population size is a parameter related with the performance of GA (number of function evaluations) being necessary its correct assignment. The following paragraphs describe the proposal to compute the optimal population size for a given optimization problem.

Based on the learning models generated by PCM, we can compute the population size necessary to solve a particular optimization problem. The idea behind this proposal is the parameter associated to the learning models: population size n. If models are capable of determining the level of hardness associated to certain population sizes, we can construct a procedure to determine the optimal population size where the problem is categorised as easy problem. The procedure to Recommend Population Size (RPS) for GA (Algorithm 4) recommend an optimal population size for a given optimization problem, the recommendation is based on the capacity of models to predict the hardness of optimization problems.

Basically, for a given optimization problem f, PCM finds the model M_n where f is categorized as easy problem (see Equation 5.2), then, the population size n related with the model M will be the population size recommend by the procedure.

Algorithm 4

```
RPS(f)
1  FOR n = 50 to 500 step 50
2    dataset = DATASET(n) //according to Algorithm 2.
3    M_n = RF_Ensemble(dataset) //according to Algorithm 3.
4    IF M_n(f) = easy
5        THEN return n
6   return undef
```

RPS has as input parameter f which is an optimization problem, 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).

According to the Algorithm 4, the optimization problem is tested through different models; to construct the models, it is created its dataset associated to a particular population size. Then the problem is passed to the model and tested, if the result is easy, then the associated population size n is returned. 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.

5.3. Summary

This chapter stated the proposal to perform the performance prediction of GA. The predictions are based on learning models, the models use as predictor variables, fitness landscape features, and as target variables, performance values of GA. Based on the capabilities of models, this contribution proposes a Procedure to Recommend Population Size; for a given optimization problem, the procedure proposes a minimum population size that solve efficiently the problem.

Chapter 6

Experimental Results

This chapter presents the main experimental results of this contribution; the first part describes the optimization problems used in the experiments, and the parameters settings for both, Fitness Landscape metrics and for the GA metaheuristic. The second part shows the accuracy and confusion matrices obtained by the learning models. Finally, it presents the results of FLA tests applied to RPS. Section 6.1 defines the problems used in the experiments, Section 6.2 shows the parameter setting used by the metrics and GA, Section 6.2.1 shows the main results obtained by the models, Section 6.4 presents the results obtained by the Procedure to Recommend Population Size), and Section 6.5 presents a study case of the usage of models.

6.1. Benchmark Problems

Generally in the majority of works, researchers use a small set of optimization problems, in the majority of cases are benchmark optimization functions [Xin09, Pitzer10, Malan09]. To avoid possible biases in computation, we consider a set of 110 benchmark optimization functions, the functions contain different characteristics and in all the cases the problem consist in minimization task. Details about its definition in Appendix A.1, and details about its general characteristics in Appendix A.2.

6.2. Parameters Settings

Table 6.1 shows the main parameters settings used to compute the Fitness Landscape metrics (defined in Section 4).

Table 6.1: Parameters of Fitness Landscape metrics.

FL metric	**Parameter-value**
Neutrality	$\delta = d_E(L_b, U_b) \times 0.1$, $\gamma = \lvert f^{max} - f^{min} \rvert \times 0.001$
Ruggedness	$\gamma = \lvert f^{max} - f^{min} \rvert \times 0.001$
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^{min} are the maximum and minimum fitness values, respectively, of functions $f \in F$ within the search space, L_u and U_b are the lower and upper bounds of its search space. All the FL metrics use a sample of $1,000$ points picked at random using uniform distribution within each problem's domain.

GA performances were obtained varying the population size and precision; Table 6.2 shows the parameters setting used by GA.

Table 6.2: 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

We set the crossover and mutation rate to 70 % and 30 % and number of generations

to 1000, and we are focusing only in two degrees of freedom: population size and precision. To get a statistically significant measure, the experiments were repeated 100 times and the mean was reported.

6.2.1. Learning Models

Although several learning methods were tested, we are only reporting results obtained by *Random Forests* [Breiman01], since this was the most accurate method. This work uses the *WEKA* framework (using 10 trees, and descriptive and dynamic features) [Hall09] 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 $n \in \mathcal{N}$. We considered the average performance as 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.

The following Tables 6.3 and 6.4 show the error (Root Mean Squared Error RMSE) and accuracy (Correctly Classified Instances) of regression and classification models, respectively; the first column is the population size (n), considered to form the models M_n, the last three columns are the RMSE and accuracy obtained by the derived models using three sets of metrics (descriptive, dynamic, and all). To train and validate the classification models, a 10-fold cross-validation was used. The fittest models are highlighted in boldface.

Table 6.3: RMSE of regression models varying the population size (n) and using different predictor variables: descriptives, dynamics, and all metrics.

	Fitness Landscape Metrics		
n	**Descriptive**	**Dynamic**	**All**
50	0.38	0.37	**0.32**
100	0.37	0.38	**0.33**
150	0.39	0.40	**0.36**
200	0.40	0.41	**0.36**
250	0.37	0.40	**0.33**
300	0.40	0.41	**0.34**
350	0.40	0.41	**0.34**
400	0.41	0.42	**0.37**
450	0.41	0.41	**0.35**
500	0.41	0.41	**0.36**
Mean	0.39	0.40	**0.35**

Table 6.3 shows high levels of error for the regression models, obtaining the best model 0.32 of error.

Table 6.4: Accuracy of classification models varying the population size (n) and using different predictor variables: descriptives, dynamics, 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	68.3 %	68.3 %	**71.8** %

Table 6.4 shows that the most accurate models are those based on both descriptive and dynamic FL metrics. According to the table, apparently, the population size is uncorrelated to the accuracy of models, this fact can be an indication about the inbalance of dataset, that is, for some cases there are more instances of one class than the other

class. E.g. the confusion matrix (Table 6.6) for the model generated from population size ($n = 500$) there are more instances classified as easy than instances classified as difficult. To measure the quality of models, Tables 6.5 and 6.6 show the confusion matrices for the most accurate and worst models ($n = 50$ and $n = 500$ respectively) in both cases using all FL metrics.

Table 6.5: Confusion matrix for the most accurate 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 6.6: Confusion matrix for the worst 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 6.5 and 6.6 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 indicate that 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%.

6.3. Random Forest Models

The models are based on the Random Forest technique implemented on WEKA framework [Hall09]. To construct the models, we use the set of predictor variables specified in Table 6.4 without a pre-selecting phase (selecting the best variables); to ensure an optimal

number of tree in each model, we tested different number of trees in each model (100 trees, 50 trees, and 10 trees) and we observed a minimal accuracy difference between models, and finally we decided 10 trees for each model. To schematize a Random Forest model, the following Figure 6.1 shows one tree of the model M_{50}.

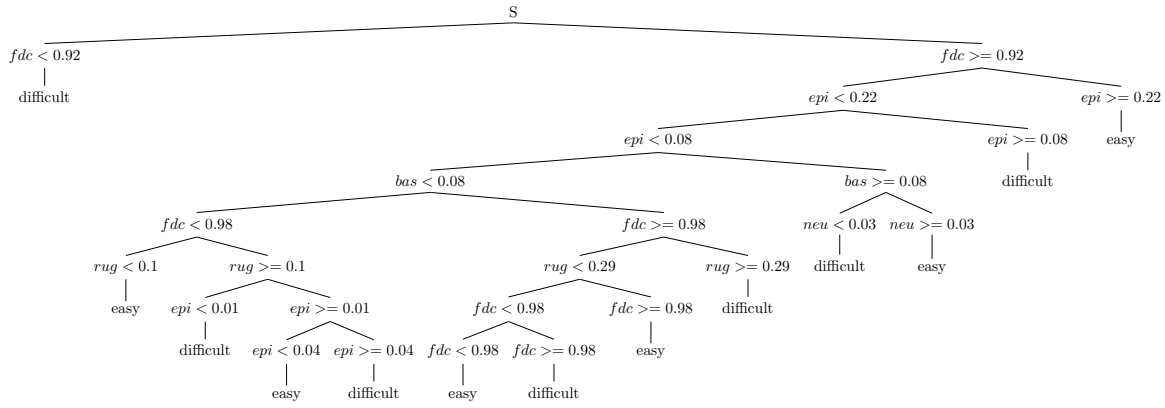

Figure 6.1: M_{50}, Random Tree 1

As we can see the model presents a set of test for the FL features (attributes) where each branch represents the outcome of the test and each leaf represents a class label (easy or difficult). When a new optimization problem arrives to the model the FL measures are tested to finally establish its difficulty. One example of the Random Forest models (M_{50}) is presented in Appendix B.

6.4. Recommend Population Size

To validate the procedure to Recommend Population Size (RPS) (Algorithm 4) we compare the optimal population size against the population size recommended by RPS, both values were obtained from the set of 110 benchmark optimization functions mentioned above. We called optimal population size to such population size where its related average performance (discretized in easy or difficult according to Equation 5.4) is classified as easy, in other words, the smallest populations where its average performance is categorized as easy. E.g., given the function f and its pairs of (population size, discretized performance): (50,difficult), (100,easy), (150, easy), etc., in this case the optimal population size will be

100 due that population size is the lowest where the problem was classified as easy. Figure 6.2 shows a scatter plot between the optimal population size against the recommended population size by *RPS* for the F set.

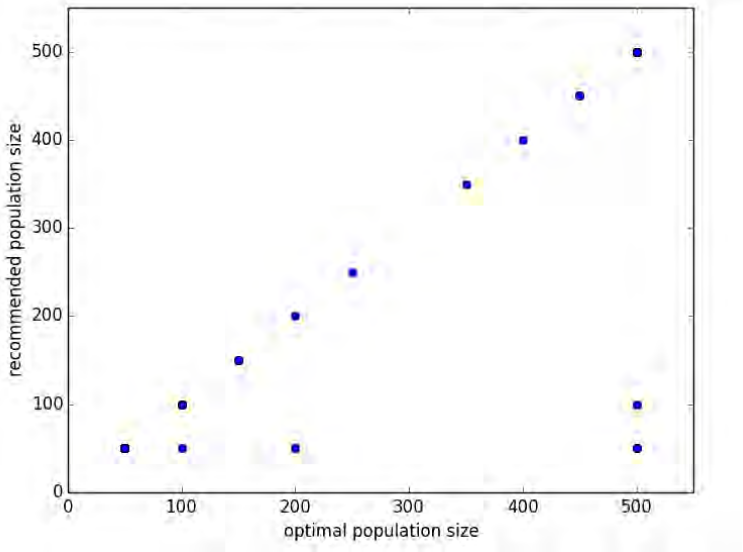

Figure 6.2: Population sizes recommended by the *RPS* procedure.

Points in Figure 6.2 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.

6.5. Study Case

This section presents an empirical study to characterize and predict the minimum population size required by Real-Coded Genetic Algorithms to solve three different instances of the University Course Timetabling Problem (UCTP). We use the Random Forest models described before, the models are based on optimization function on the continuous domain in two dimensions; the UCTP instances presented here are on the continuous domain in four dimensions. Despite the difference of dimensions between the models and the UCTP

instances, the models require as input parameters only the FL features of problems, being the models able to predict the hardness of problems in different dimensions. To perform the characterization a FLA was performed on the UCTP instances using the landscape features described in Chapter 4. Once the FL were computed, the RPS procedure predicted the optimal population required by GA to solve the instances.

6.5.1. University Course Timetabling Problem

The University Course Timetabling Problem (UCTP) is a combinatorial problem whose general objective is to find the best combination of resources covering certain needs and satisfying a set of constraints. Resources are represented by lecturers, classrooms, and time slots, among others, and needs, generally, are the academic courses offered by the university.

The UCTP can be defined as follows. Given the following preliminary definitions: $\mathbb{C}$ is a set of courses $c_i \in \mathbb{C}$, $i \in [1, N_{\mathbb{C}}]$, $\mathbb{L}$ is a set of lecturers $l_j \in \mathbb{L}$, $j \in [1, N_{\mathbb{L}}]$, $\mathbb{R}$ is set of classrooms $r_k \in \mathbb{R}$, $k \in [1, N_{\mathbb{R}}]$, $\mathbb{T}_1$ is a set of available times in the week (from monday to thursday) $a_l \in \mathbb{T}_1$, $l \in [1, N_{\mathbb{T}_1}]$, $\mathbb{T}_2$ is a set of available times on friday $b_m \in \mathbb{T}_2$, $[m \in 0, 1, ..., N_{\mathbb{T}_2}]$. All the indices start at 1, except the index for $\mathbb{T}_2$ which starts at 0, to simplify the cases when a course does not occur in a given day (i.e., if does not have an assigned time).

Constraints are represented by the following functions:

- $\mathbb{H}(\cdot)$ is a function that returns the penalization for the hard constraints, constraints are: 1) a classroom cannot be assigned to more than one course at the same time/day, or a lecturer cannot be assigned to more than one course at the same time/day, 2) the number of weekly hours assigned to a course must match the course's needs.

- $\mathbb{S}(\cdot)$ is a function that returns the penalization for the soft constraints, constraints

are: 1) classrooms must be assigned consecutively (no holes in schedule), 2) check the suitable classroom (theory or practice), 3) chek if teacher not pass his/her assignment hours 4) the lecturer profile must match the course requirement, 5) the assigned classrooms must satisfy the needs of course capacity, and 6) check the preference of lecturers's time,

The objective is to minimize the penalization of hard constraints ($\mathbb{H}$) and soft constraints ($\mathbb{S}$); the hard constraints must be fulfilled and the soft constraints must be minimized [Abdullah12]. A mathematical formulation can be represented by:

Determine an assignment $[X_{l,r,t_1,t_2}]_{N_C}$ that minimizes:

$$f(X) = \mathbb{H}(X) + \mathbb{S}(X) \tag{6.1}$$

Subject to:

$$x \in [a, b] \text{ is the interval, } a \leq x \leq b \tag{6.2}$$

$$\forall j \in [1, N_L], \forall k \in [1, N_R], \forall l \in [1, N_{T_1}], \forall m \in [1, N_{T_2}], \sum_{c \in C} \mathcal{H}(x_{j,k,l,m}) = 0 \tag{6.3}$$

$f(X)$ computes the total penalization for a given X (Equation 6.1), Equation (6.2) checks the interval values for each dimension, the soft constraints can be violated, and Equation (6.3) prevents the hard constraint violations for all $c \in C$. Details of the solution of UCTP using GA can be found in [Rodriguez-Maya14b].

6.5.2. UCTP Instances

The characterization and prediction was performed on three UCTP instances: Instituto Tecnologico de Zitacuaro (ITZ), Instituto Tecnologico del Valle de Morelia (ITVM), and Instituto Tecnologico de Tuxtla Gutierrez (ITTG). Table 6.7 shows the sets of tasks and resources for each university.

Table 6.7: Tasks and resources available for each UCTP instance.

Task	**ITZ**	**ITVM**	**ITTG**
Courses ($\mathbb{C}$)	181	112	180
Lecturers ($\mathbb{L}$)	69	122	218
Classrooms ($\mathbb{R}$)	34	38	102
Available times per week, from monday to thursday ($\mathbb{T}_1$)	12	12	12
Available times on friday ($\mathbb{T}_2$)	12	12	12

6.5.3. Characterization and Prediction

The parameters used by the FL metrics are the same that Table 6.1. A numerical characterization of the UCTP instances is shown in Table 6.8.

Table 6.8: FLA on the UCTP instances.

	FL metrics					
Instance	**Ruggedness**	**Neutrality**	**Epistasis**	**B. of attraction**	**NSC**	**FDC**
ITZ	0.835	0.422	0.0	0.001	-0.871	0.941
ITVM	0.872	0.321	0.0	0.217	-1.832	0.862
ITTG	0.860	0.362	0.0	0.25	-2.400	0.905

In general, all the instances present similar values of ruggedness and neutrality: the landscape is rugged with some neutral areas. Due to the nature of fitness function (separable function) all the instances have the same epistasis. On the other hand, according to NSC the ITTG is most difficult, while ITZ the easiest. FDC determines that instances are not deceptive.

To perform the population size recommendation, RPS uses the FL features calculated in Table 6.8 and the experiments were repeated 10 times. Table 6.9 shows the population size recommended by the RPS on the UCTP instances.

Table 6.9: Recommended population size for the UCTP instances.

Instance	**min**	**max**	**avg**	σ
ITZ	100	200	140.0	51.63
ITVM	150	500	315.0	177.09
ITTG	100	500	200.0	118.90

According to results, in average, the ITVM instance needs a larger population size than the other instances, this is an indication about hardness of instances: the more population size, the more difficult to solve. On the other hand, the ITZ instance should be the most difficult instance due to it contains a major number of needs (courses) to be covered and minus resources (lecturers and classrooms) to be assigned (according to Table 6.7). However, the ITZ appears as the easiest instance, maybe from its landscape form (low level of basins of attraction) can give us some insights about its difficulty.

6.6. Summary

This chapter presents the main results of this work. The models are based on descriptive and dynamic features; the most accurate models were the models based on a mixture of descriptive and dynamic features, and models based only in dynamic features. A direct application of models was the Procedure to Recommend Population Size (RPS), for a given optimization problem. RPS recommends the minimal population size for an optimal solution for that problem. A study case of application of FLA to characterize and predict the difficulty of some instances of University Course Timetabling Problem was presented.

Chapter 7

Conclusions and Future Work

7.1. Conclusions

Generally, many approaches use only Fitness Landscape metrics in isolation to predict the hardness of Evolutionary Algorithms when solving optimization problems, being the results not very persuasive in the majority of cases. The performance of some metaheuristics is closely related to the geometric form of the problem's landscape, e.g., rugged landscapes generally are difficult to solve. What happens when a landscape contain low levels of ruggedness and high levels of deceptiveness?, surely users could categorize such problems, as easy. However, some deceptive problems are categorized as hard problems. To perform an accurate categorization (characterization) it is necessary to use a set of FL metrics that capture the majority of aspects of optimization problems: multimodality, neutrality, separability, evolvability, etc.

This contribution presents a procedure called Performance Classification Models (PCM) to construct learning models to predict the difficulty of GA in the solution of continuous optimization problems in two dimensions. PCM uses a set of Fitness Landscape metrics to perform a Fitness Landscape Analysis; these metrics are grouped into descriptive and dynamic; while descriptive metrics measure the descriptive properties of optimization problems, the dynamic metrics capture the evolvability of the heuristic. The models we-

re based on a supervised machine learning technique: *Random Forests*. The models used Fitness Landscape features as predictor variables and the performance exhibited by GA, expressed as success rate, as target variable, and, the experiments were performed on on 110 continuous optimization problems in two dimensions.

It is important to remark that given a new optimization problem, the generated models can predict the performance of GA to solve it (easy or difficult), based uniquely on the problem's FL-metrics. The models that used all the 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 behaviour 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. To support the experimental results, this work used a set of 110 optimization problems, contrary to the more limited number of benchmark functions reported in literature.

The hardness of GA to solve optimization problems is approximated with the performance of GA to solve them. In this context, performance is defined as the rate of successful trials, a trial is sucessful when the global optimum is reached. Literature reports other types of performances as the number of function evaluations, the expected running time, etc. In this work it is used the sucess rate because it is closer to the natural aim of metaheuristics (to get the global optimum) and because the result is a number between $[0, 1]$. Future works, will develop models using other performance measures.

Another proposal based on this work, is the procedure to recommend population size (RPS). The proposed procedure uses the set of models to try to determine the smallest population size for GA; the recommendation is based on a population where the problem is classified as easy. The first results showed a strong correlation between the optimal popu-

lation size and the recommended population size, having a correlation value equal to 0.92 and an accuracy of prediction of 95%. RPS was tested on a set of instances of the University Course Timetabling Problem (UCTP). Instances are based on mexican universities which have different needs and resources. The results showed how RPS can predict optimal population size in different domains; the methodology to develop the learning models and RPS can be extended to other dimensions and other domains.

7.2. Future Work

This work represent an effort to solve a particular instance of the Algorithm Selection Problem (ASP) [Rice76]: the Genetic Algorithms case on the continuous domain. The original Rice's model, establishes, for a given set of problems and its corresponding best features, and a set of algorithms, he proposes the use of performance to select the best algorithm to solve a particular problem. In this context, the performance prediction is very important to support the selection process. For continuous domains, the solution requires mainly the comprehension of: the search space through sampling, the optimization problems, the algorithms, and performance. The following tasks are necessaries to extend this work in the solution of other ASP aspects.

1. The study and use of different sampling methods. The analysis of the search space, specially on continuous domains, requires the use of efficient sampling methods to obtain the more representative configurations. For instance, the Latin hypercube sampling (LHS) has been tested on different scenarios as an efficient sampling method; LHS generates points, following a probability density function.

2. The development and use of other FL metrics. The metrics must capture the most important features of optimization problems, the most representative features can be selected to be incorporated as predictive variables into the learning models. For instance, the usage of more innovative FL metrics as: Accumulated escape probability [Lu11], the Information Content for continuous landscapes [Muñoz15], the Exploratory Landscape Analysis (Low-Level Features) [Mersmann11], Cell Mapping Techniques

[Kerschke14], among others.

3. The incorporation of problems on higher dimensions. It is clear that the behavior of problems in higher dimensions is very different that problems in lower dimensions, and many real life applications are in higher dimensions. It is very important the study of the most representative benchmark problems in higher dimensions. For instance, some of the benchmark problems implemented in this work can be easily tested in higher dimensions.

4. The implementation of finer classification, and/or regression. One of the most important aims of ASP is the prediction of performance; the output of learning models must be too close to the objective, that is, ideally, the prediction must be a regression task.

5. The incorporation of other Evolutionary Algorithms. The original ASP model contemplates many different algorithms, those algorithms contain different characteristics and behaviors, then, it is necessary the implementation of different algorithms and the establishment of its main features to solve another ASP instances.

6. Propose a solution of the ASP problem, and a input-parameter selector for metaheuristics. Once the above works have been resolved, can be proposed a possible solution to the ASP. The solution must contain the basic elements (set of problems and its features, set of algorithms and the corresponding performances) to create regression learning models. The learning models can consider into the predictor variables, the algorithm's input parameters, and the user can select the best input parameters based on its performance.

Appendix A

Optimization Problems

A.1. Benchmark Functions

The following table shows the definition of functions used in this work, more details in [Jamil13].

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	$f(x_1,x_2) = \frac{x_1^2}{4000} + \frac{x_2^2}{4000} - \cos(x_1)\cos\left(\frac{x_2}{\sqrt{2}}\right) + 1$	$-600 \leq x_i \leq 600, i \in [1,2]$	$f(\vec{x}^*) = 0, \vec{x}^* = (0,0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(\vec{x}) = \sum_{i=0}^{n-1} x_i^2$	$-5.12 \leq x_i \leq 5.12, i \in [1, 2]$	$f(\vec{x}^*) = 0, \vec{x}^* = (0, 0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{16}	Trid	$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2 - x_1 x_2 + 1$	$-4 \leq x_i \leq 4, i \in [1, 2]$	$f(\vec{x}^*) = -1$, $\vec{x}^* = (2, 2)$
f_{17}	Zakharov	$f(x_1, x_2) = (0.5x_1 + 1.x_2)^4 + (0.5x_1 + 1.x_2)^2 + x_1^2 + x_2^2$	$-5 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0, 0)$
f_{18}	Dropwave	$f(x_1, x_2) = \frac{-\cos\left(12\sqrt{x_1^2+x_2^2}\right)-1}{0.5\left(x_1^2+x_2^2\right)+2}$	$-5.12 \leq x_i \leq 5.12, i \in [1, 2]$	$f(\vec{x}^*) = -1$, $\vec{x}^* = (0, 0)$
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	$f(x_1, x_2) = 0.5(x_1^4 - 16x_1^2 + 5x_1 + x_2^4 - 16x_2^2 + 5x_2)$	$-5 \leq x_i \leq 5, i \in [1, 2]$	$f(\vec{x}^*) = -39.16599 * 2$, $\vec{x}^* = (-2.903534, -2.903534)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{23}	Randompeaks	$f(x_1,x_2) = -2e^{-0.5\left((x_1-21)^2+(x_2-25)^2\right)} - 2e^{-0.5\left((x_1-8)^2+(x_2-25)^2\right)} + 5e^{-0.1\left((x_1-15)^2+(x_2-20)^2\right)} + 2e^{-0.5\left((x_1-25)^2+(x_2-16)^2\right)} - 2e^{-0.08\left((x_1-20)^2+(x_2-15)^2\right)} + 2e^{-0.5\left((x_1-5)^2+(x_2-14)^2\right)} + 3e^{-0.08\left((x_1-25)^2+(x_2-10)^2\right)} + 2e^{-0.1\left((x_1-10)^2+(x_2-10)^2\right)} - 2e^{-0.5\left((x_1-5)^2+(x_2-10)^2\right)} - 4e^{-0.1\left((x_1-15)^2+(x_2-5)^2\right)}$	$0 \leq x_i \leq 30, i \in [1,2]$	$f(\vec{x}^*) = -3.98654$, $\vec{x}^* = (15.01369, 4.9643)$
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	$f(x_1,x_2) = \frac{1}{16}(x_1-1)^2(10\sin^2(\pi(\frac{1}{4}(x_1-1)+1)+1)+1) + \sin^2(\pi(\frac{x_1-1}{4}+1)) + \frac{1}{16}(x_2-1)^2(\sin^2(2\pi(\frac{x_2-1}{4}+1))+1)$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1,1)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{26}	Dejong	$a = [[-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32],[-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32]]$ $f(x_1,x_2) = (0.002 + \sum_{i=1}^{25} \frac{1}{i+(x_1-a_{1i})^6+(x_2-a_{2i})^6})^{-1}$	$-65.536 \leq x_i \leq 65.536, i \in [1,2]$	$f(\vec{x}^*) = 0.7300895679837434 2$, $\vec{x}^* = (-31.97855, -31.97855)$
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	$f(x_1,x_2) = \frac{\sin^2(x_1^2-x_2^2)-0.5}{(0.001(x_1^2-x_2^2)+1)^2} + 0.5$	$-100 \leq x_i \leq 100, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (-0.231665, 0.232741)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1, x_2) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0,\ \vec{x}^* = (0,0)$
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	$f(x_1, x_2) = 0.01\lvert x_1 + 10\rvert + 100\sqrt{\lvert x_2 - 0.01x_1^2\rvert}$	$-15 \leq x_i \leq 3, i \in [1,2]$	$f(\vec{x}^*) = 0,\ \vec{x}^* = (-10, 1)$
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	$f(x_1, x_2) = -200e^{-0.02\sqrt{x_1^2+x_2^2}}$	$-32 \leq x_i \leq 32, i \in [1,2]$	$f(\vec{x}^*) = -200,$ $\vec{x}^* = (0,0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
f_{38}	Ackley3	$f(x_1, x_2) = 5e^{\sin(3x_2)+\cos(3x_1)} - 200e^{-0.02\sqrt{x_1^2+x_2^2}}$	$-32 \leq x_i \leq 32, i \in [1, 2]$	$f(\vec{x}^*) = -195.629$, $\vec{x}^* = (-0.682577, -0.360702)$
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	$f(x_1, x_2) = \sum_{i=1}^{10}(\exp(-0.1 * i * x_1) - 5 * \exp(-0.1 * i * x_2) - \exp(-0.1 * i) - 5 * \exp(10 * 0.1 * i))^2$	$0 \leq x_i \leq 20, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (1, 10)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1, x_2) = (x_1^2)^{(x_2^2+1)} + (x_2^2)^{(x_1^2+1)}$	$-1 \leq x_i \leq 4, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0, 0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1,x_2) = 2x_1^2 - 1.05x_1^4 + x_1^6/6 + x_1x_2 + x_2^2$	$-5 \leq x_i \leq 5, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0,0)$
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	$f(x_1,x_2) = x_1^2 - 12x_1 + 11 + 10\cos((\pi x_1)/2) + 8\sin((5\pi x_1)/2) - (1/5)^{0.5}\exp(-0.5(x_2-0.5)^2)$	$-30 \leq x_i \leq 30, i \in [1,2]$	$f(\vec{x}^*) = -42.49717342349103$, $\vec{x}^* = (6.18987, 0.75477)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1, x_2) = 10^5 * x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^-5(x_1^2 + x_2^2)^4$	$-20 \leq x_i \leq 20, i \in [1,2]$	$f(\vec{x}^*) = -24771.093749999996$, $\vec{x}^* = (0, -15)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1,x_2) = (x_1 - 13 + ((5 - x_2)x_2 - 2)x_2)^2 + (x_1 - 29 + ((x_2 + 1)x_2 - 14)x_2)^2$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (5,4)$
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	$f(x_1,x_2) = \sum_{i=0}^{4}(i+1)\cos(ix_1+i+1)\sum_{j=0}^{4}(j+1)\cos((j+2)x_2 + j + 1$	$-10 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = -176.542$, $\vec{x}^* = (-7.589893, -7.708314)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1, x_2) = \sqrt{\lvert \cos(\sqrt{\lvert x_1^2 + x_2^2 \rvert}) \rvert} + 0.01(x_1 + x_2)$	$-1 \leq x_i \leq 1, i \in [1, 2]$	$f(\vec{x}^*) = 0.3748970685702472$, $\vec{x}^* = (3.26783, -3.0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1, x_2) = 0.001(\lvert x_1^10 - 20x_1^9 + 180x_1^8 - 960x_1^7 + 3360x_1^6 - 8064x_1^5 + 1334x_1^4 - 15360x_1^3 + 11520x_1^2 - 5120x_1 + 2624 \rvert \lvert x_2^4 + 12x_2^3 + 54x_2^2 + 108x_2 + 81 \rvert)^2$	$-10 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = 0,\ \vec{x}^* = (2, -3)$
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	$f(x_1, x_2) = 1 + \sin(x_1)^2 + \sin(x_2)^2 - 0.1 \exp[-(x_1^2 + x_2^2)]$	$-10 \leq x_i \leq 10, i \in [1, 2]$	$f(\vec{x}^*) = 0.9,\ \vec{x}^* = (0, 0)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1,x_2) = -3803.84 - 138.08x_1 - 232.92x_2 + 128.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2$	$-1 \leq x_i \leq 10, i \in [1,2]$	$f(\vec{x}^*) = -3873.7241821830326,\ \vec{x}^* = (0.19388, 0.48513)$

No.	Name	Function	Lower and Upper bounds	Global Optima
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	$f(x_1, x_2) = -x_1 x_2 (72 - 2x_1 - 2x_2)$	$0 \leq x_i \leq 500, i \in [1, 2]$	$f(\vec{x}^*) = -3456$, $\vec{x}^* = (12, 12)$
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	$f(x_1, x_2) = x_1^4 - 4x_1^3 + 4x_1 x_2^2$	$-5 \leq x_i \leq 5, i \in [1, 2]$	$f(\vec{x}^*) = 0$, $\vec{x}^* = (0, 0)$

No.	**Name**	**Function**	**Lower and Upper bounds**	**Global Optima**
f_{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	$f(\vec{x}) = 1 + \sum_{i=1}^{2} 8(\sin(7(x_i - 0.9)^2))^2 + 6(\sin(14(x_1 - 0.9)))^2 + (x_i - 0.9)^2$	$-500 \leq x_i \leq 500, i \in [1, 2]$	$f(\vec{x}^*) = 1$, $\vec{x}^* = (0.9, 0.9)$

A.2. Features of Functions

Table A.2 indicates two of the most representative features: separability[1] and multimodality[2] [Jamil13].

Table A.2: Main features of functions.

No.	Function	Features	No.	Function	Features
1	Ackley	Non-separable, multimodal	56	Chen Bird	Non-separable, multimodal
2	Beale	Non-separable, unimodal	57	Chen v	Non-separable, multimodal
3	Bohachevsky	Separable, multimodal	58	Chichinadze	Separable, multimodal
4	Booth	Non-separable, unimodal	59	Chung Reynolds	Partially-separable, unimodal
5	Branin	Non-separable, multimodal	60	Cosine mixture	Separable, multimodal
6	Dixon Price	Non-separable, unimodal	61	Csendes	Separable, multimodal
7	Goldstein Price	Non-separable, multimodal	62	Cube	Non-separable, unimodal
8	Griewank	Non-separable, multimodal	63	Damavandi	Non-separable, multimodal
9	Hump	Separable, unimodal	64	Deckkers Aarts	Non-separable, multimodal
10	Michalewicz	Non-separable, multimodal	65	El Attar Vidyasagar Dutta	Non-separable, unimodal

[1] If variables in a function are independent, then the function is separable.
[2] Multimodality refers to such functions that have multiple local optima.

Table A.2: Main features of functions.

No.	**Function**	**Features**	**No.**	**Function**	**Features**
11	Rastrigin	Non-separable, multimodal	66	Egg Crate	Separable
12	Rosenbrock	Non-separable, unimodal	67	Exponential	Non-separable, multimodal
13	Schwefel	Partially-separable, unimodal	68	Exp2	Separable
14	Shubert	Separable, multimodal	69	Freudenstein Roth	Non-separable, multimodal
15	Sphere	Separable, unimodal	70	Giunta	Separable, multimodal
16	Trid	Non-separable, multimodal	71	Hansen	Separable, multimodal
17	Zakharov	Non-separable, multimodal	72	Hosaki	Non-separable, multimodal
18	Dropwave	Non-separable, multimodal	73	Jennrich Sampson	Non-separable, multimodal
19	Egg Holder	Non-separable, multimodal	74	Keane	Non-separable, multimodal
20	Holder	Non-separable, multimodal	75	Leon	Non-separable, Unimodal
21	Levy13	Non-separable, multimodal	76	Mccormick	Non-separable, multimodal
22	Styblinski Tang	Non-separable, multimodal	77	Mishra3	Non-separable, multimodal
23	Random peaks	Non-separable, multimodal	78	Mishra4	Non-separable, multimodal

Table A.2: Main features of functions.

No.	**Function**	**Features**	**No.**	**Function**	**Features**
24	Sum of different power	Non-separable, unimodal	79	Mishra5	Non-separable, multimodal
25	Levy	Non-separable, multimodal	80	Mishra6	Non-separable, multimodal
26	Dejong	Non-separable, multimodal	81	Mishra8	Non-separable, multimodal
27	Langermann	Non-separable, multimodal	82	Pen Holder	Non-separable, multimodal
28	Himmelblau	Non-separable, multimodal	83	Pathological	Non-separable, multimodal
29	Sum squares	Separable, unimodal	84	Periodic	Separable
30	Schaffer2	Non-separable, unimodal	85	Powell sum	Non-separable, Unimodal
31	Easom	Separable, multimodal	86	Price1	Separable, multimodal
32	Matyas	Non-separable, unimodal	87	Price2	Non-separable, multimodal
33	Cross in Tray	Non-separable, multimodal	88	Price3	Non-separable, multimodal
34	Bukin	Non-separable, multimodal	89	Price4	Non-separable, multimodal
35	Schaffer4	Non-separable, unimodal	90	Qing	Separable, multimodal
36	Equal peaks	Separable	91	Quadratic	Non-separable
37	Ackley2	Non-separable, unimodal	92	Quartic	Separable

Table A.2: Main features of functions.

No.	**Function**	**Features**	**No.**	**Function**	**Features**
38	Ackley3	Non-separable, unimodal	93	Quintic	Separable, multimodal
39	Ackley4	Non-separable, multimodal	94	Rosenbrock modified	Non-separable, multimodal
40	Adjiman	Non-separable, multimodal	95	Rotated ellipse	Non-separable, unimodal
41	Alpine1	Separable, multimodal	96	Rotated ellipse2	Non-separable, multimodal
42	Alpine2	Separable, multimodal	97	Rump	Non-separable, unimodal
43	Bartels	Non-separable, multimodal	98	Salomon	Non-separable, multimodal
44	Bigg exp2	Non-separable, multimodal	99	Sargan	Non-separable, multimodal
45	Bird	Non-separable, multimodal	100	Schaffer3	Non-separable, unimodal
46	Bohachevsky2	Non-separable, multimodal	101	Schumer Steiglitz	Separable, unimodal
47	Bohachevsky3	Non-separable, multimodal	102	Schwefel24	Separable, multimodal
48	Branin Rcos	Non-separable, multimodal	103	Schwefel222	Non-separable, unimodal
49	Branin Rcos2	Non-separable, multimodal	104	Schwefel236	Separable, multimodal
50	Brent	Non-separable, unimodal	105	Streched v sine wave	Non-separable, unimodal

Table A.2: Main features of functions.

No.	**Function**	**Features**	**No.**	**Function**	**Features**
51	Brown	Non-separable, unimodal	106	Testtube holder	Separable, multimodal
52	Bukin2	Non-separable, multimodal	107	Trecanni	Separable, unimodal
53	Bukin4	Separable, multimodal	108	Trefethen	Non-separable, multimodal
54	Three hump camel function	Non-separable, multimodal	109	Trigonometric	Non-separable, multimodal
55	Six hump camel function	Non-separable, multimodal	110	Trigonometric2	Non-separable, multimodal

Appendix B

Models

The following are the ensemble of Random Forest trees generated by model M_{50}; the model use descriptive and dynamic fitness landscape features.

B.1. M_{50}

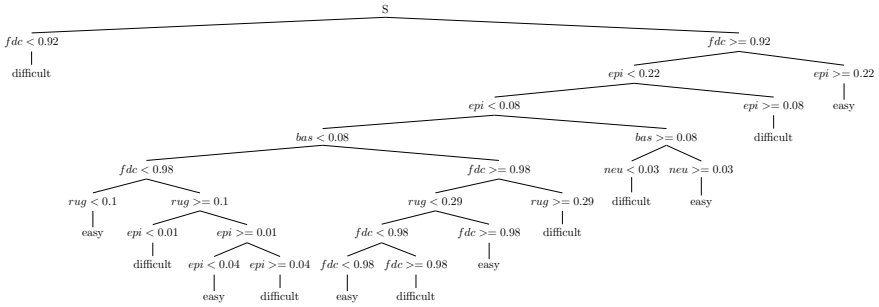

Figure B.1: M_{50}, Random Tree 1

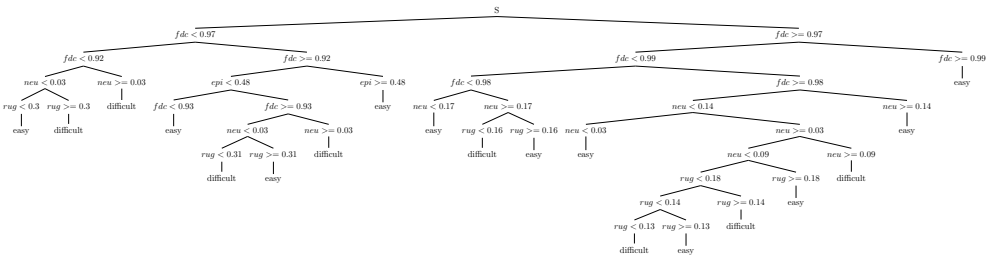

Figure B.2: M_{50}, Random Tree 2

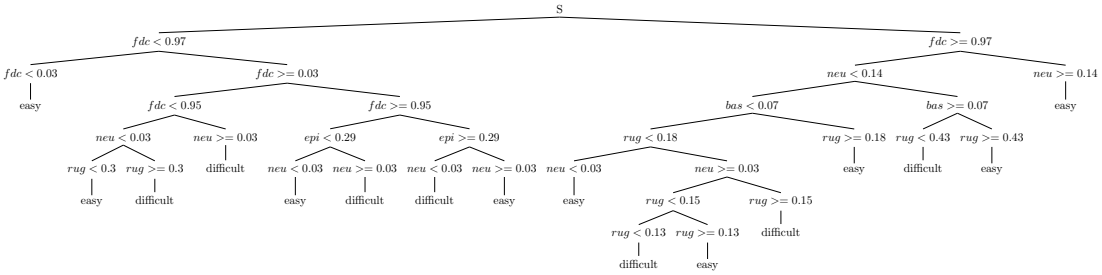

Figure B.3: M_{50}, Random Tree 3

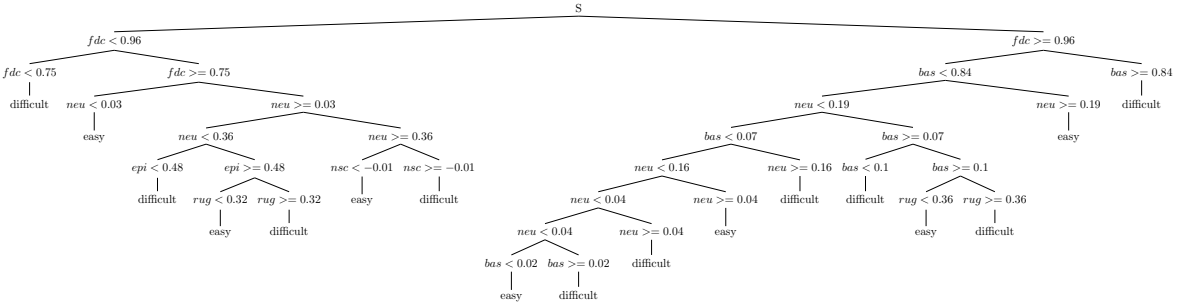

Figure B.4: M_{50}, Random Tree 4

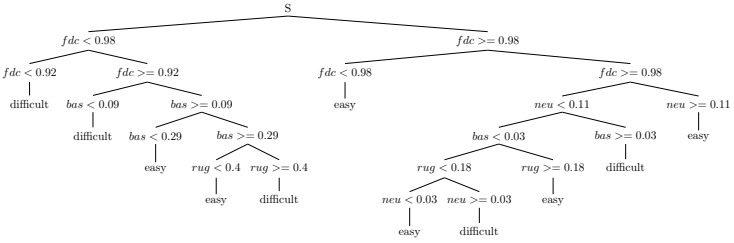

Figure B.5: M_{50}, Random Tree 5

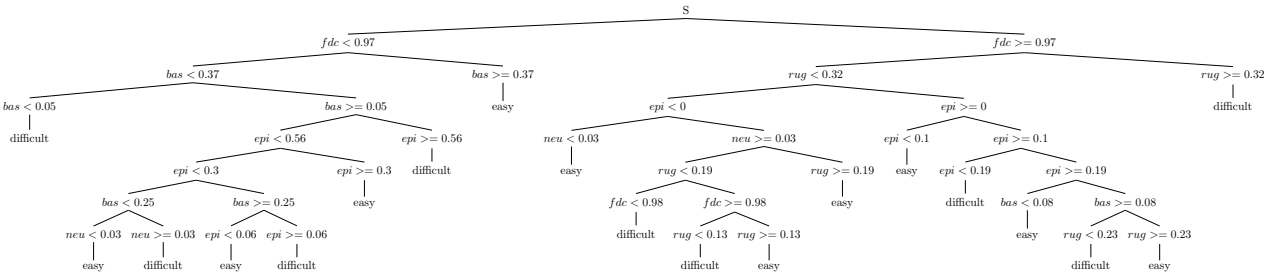

Figure B.6: M_{50}, Random Tree 6

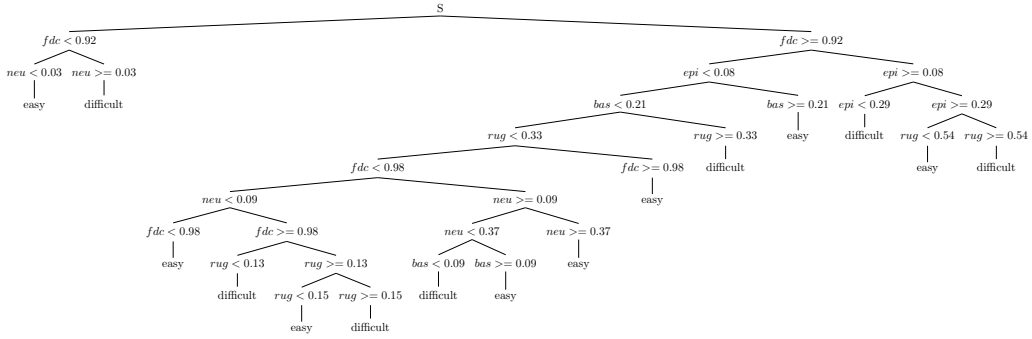

Figure B.7: M_{50}, Random Tree 7

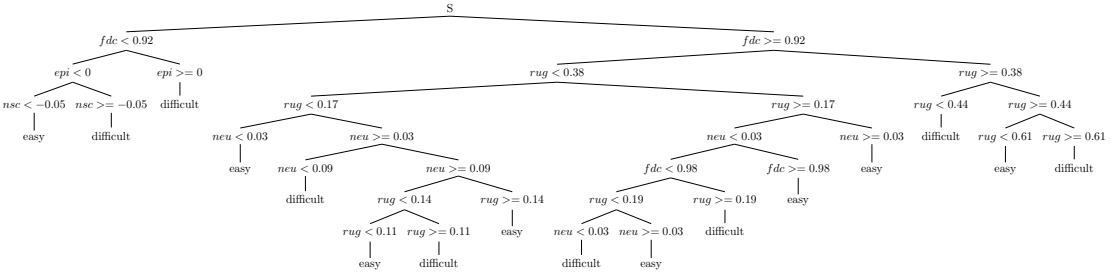

Figure B.8: M_{50}, Random Tree 8

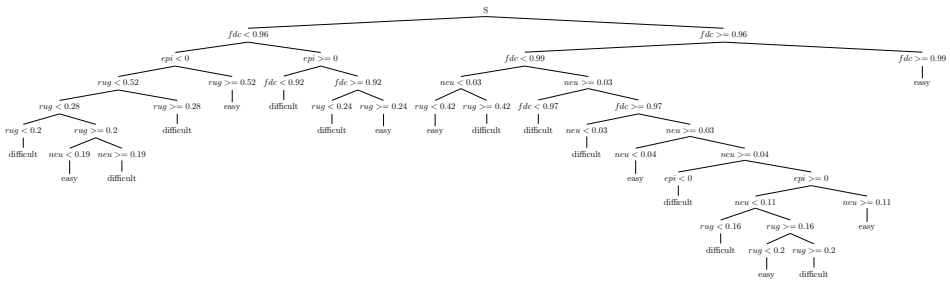

Figure B.9: M_{50}, Random Tree 9

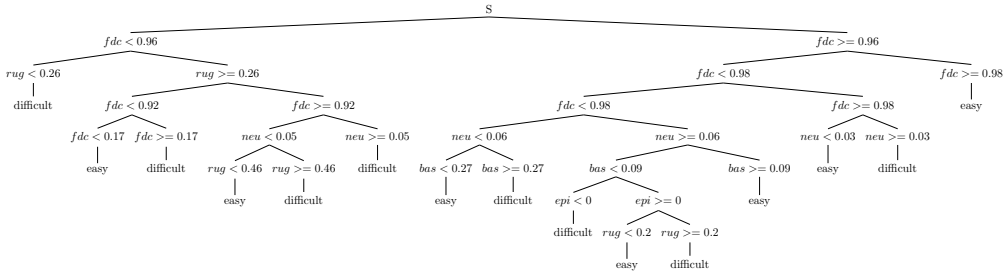

Figure B.10: M_{50}, Random Tree 10

References

[Abdullah12] Abdullah, S., Turabieh, H., McCollum, B., y McMullan, P. A hybrid metaheuristic approach to the university course timetabling problem. *J. Heuristics*, 18(1):1–23, 2012.

[Altenberg97] Altenberg, L. Fitness distance correlation analysis: An instructive counterexample. *En* T. Back, ed., *ICGA*, págs. 57–64. Morgan Kaufmann, 1997.

[Asmus14] Asmus, J., Borchmann, D., Sbalzarini, I. F., y Walther, D. Towards an fca-based recommender system for black-box optimization. *En* S. O. Kuznetsov, A. Napoli, y S. Rudolph, eds., *Proceedings of the 3rd International Workshop on "What can FCA do for Artificial Intelligence?"(FCA4AI'14)*, tomo 1257 de *CEUR Workshop Proceedings*, págs. 35–42. 2014.

[Auger12] Auger, A., Hansen, N., Heidrich-Meisner, V., Mersmann, O., Posik, P., y Preuss, M. Gecco 2012 workshop on black-box optimization benchmarking (bbob). *En GECCO 2012: Genetic and Evolutionary Computation Conference Companion*. ACM, New York, NY, USA, 2012.

[Bischl12] Bischl, B., Mersmann, O., Trautmann, H., y Preuß, M. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. *En Proceedings of the 14th Annual Conference on Genetic*

and Evolutionary Computation, GECCO '12, págs. 313–320. ACM, New York, NY, USA, 2012. ISBN 978-1-4503-1177-9.

[Blum03] Blum, C. y Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, sep. 2003. ISSN 0360-0300.

[Boussaïd13] Boussaïd, I., Lepagnot, J., y Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.*, 237:82–117, jul. 2013. ISSN 0020-0255.

[Breiman01] Breiman, L. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125.

[Caamaño10] Caamaño, P., Prieto, A., Becerra, J. A., Bellas, F., y Duro, R. J. Real-valued multimodal fitness landscape characterization for evolution. *En Proceedings of the 17th International Conference on Neural Information Processing: Theory and Algorithms - Volume Part I*, ICONIP'10, págs. 567–574. Springer-Verlag, Berlin, Heidelberg, 2010. ISBN 3-642-17536-8, 978-3-642-17536-7.

[Caamaño13] Caamaño, P., Bellas, F., Becerra, J. A., y Duro, R. J. Evolutionary algorithm characterization in real parameter optimization problems. *Appl. Soft Comput.*, 13(4):1902–1921, 2013.

[Chan03] Chan, K. Y., Aydin, M. E., y Fogarty, T. C. An epistasis measure based on the analysis of variance for the real-coded representation in genetic algorithms. *En IEEE Congress on Evolutionary Computation*, págs. 297–304. IEEE, 2003.

[Chen08] Chen, Y., Hu, J., Hirasawa, K., y Yu, S. Solving deceptive problems using a genetic algorithm with reserve selection. *En Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, págs. 884–889. 2008.

[Chipperfield94] Chipperfield, A., Fleming, P., Pohlheim, H., y Fonseca, C. Genetic algorithm toolbox for use with matlab. Inf. téc., University of Sheffield, 1994.

[Davidor90] Davidor, Y. Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383, 1990.

[Deb93] Deb, K. y Goldberg, D. E. Analyzing deception in trap functions. *Foundations of genetic algorithms*, 2:93–108, 1993.

[Galván-López06] Galván-López, E. y Poli, R. Some steps towards understanding how neutrality affects evolutionary search. *En* T. P. Runarsson, H.-G. Beyer, E. K. Burke, J. J. M. Guervós, L. D. Whitley, y X. Yao, eds., *PPSN*, tomo 4193 de *Lecture Notes in Computer Science*, págs. 778–787. Springer, 2006. ISBN 3-540-38990-3.

[Graff10] Graff, M. y Poli, R. Practical performance models of algorithms in evolutionary program induction and other domains. *Artificial Intelligence*, 174(15):1254–1276, oct. 2010. ISSN 0004-3702.

[Graff12] Graff, M., Poli, R., y Flores, J. J. Models of performance of evolutionary program induction algorithms based on indicators of problem difficulty. *Evolutionary Computation*, nov. 2012. ISSN 1063-6560.

[Graff13] Graff, M., Escalante, H. J., Cerda-Jacobo, J., y Avalos-Gonzalez, A. Models of performance of time series forecasters. *Neurocomputing*, 122:375–385, dic. 2013. ISSN 0925-2312. 00001.

[Grefenstette92] Grefenstette, J. J. Deception considered harmful. *En Foundations of Genetic Algorithms 2*, págs. 75–91. Morgan Kaufmann, 1992.

[Hall09] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., y Witten, I. H. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, nov. 2009. ISSN 1931-0145.

[Hansen10] Hansen, N., Auger, A., Finck, S., y Ros, R. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Inf. Téc. RR-7215, INRIA, Paris, France, September 2010.

[Haupt04] Haupt, R. L. y Haupt, S. E. *Practical genetic algorithms*. J. Wiley, Hoboken, N.J., 2004.

[Herrera98] Herrera, F., Lozano, M., y Verdegay, J. L. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12:265–319, 1998.

[Horn94] Horn, J. y Goldberg, D. E. Genetic algorithm difficulty and the modality of fitness landscapes. *Urbana*, 51:61801–2996, 1994.

[Horn95] Horn, J. y Goldberg, D. E. Genetic algorithm difficulty and the modality of the fitness landscapes. *En* L. D. Whitley y M. D. Vose, eds., *FOGA-3*, págs. 243–269. Morgan Kaufmann, 1995. ISBN 1-55860-356-5.

[Huang09] Huang, D., Shen, Z., Miao, C., y Leung, C. Fitness landscape analysis for resource allocation in multiuser ofdm based cognitive radio systems. *SIGMOBILE Mob. Comput. Commun. Rev.*, 13(2):26–36, sep. 2009. ISSN 1559-1662.

[Jamil13] Jamil, M. y Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *IJMNO*, 4(2):150–194, 2013.

[Jones94] Jones, T. A Model Of Landscapes. Inf. téc., Santa Fe Institute, 1994.

[Jones95a] Jones, T. *Evolutionary Algorithms, Fitness Landscapes and Search*. Tesis Doctoral, University of New Mexico, 1995.

[Jones95b] Jones, T. One operator, one landscape. Inf. téc., Santa Fe Institute, Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA, 1995.

[Jones95c] Jones, T. y Forrest, S. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. *En Proceedings of the 6th International Conference on Genetic Algorithms*, págs. 184–192. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995. ISBN 1-55860-370-0.

[Kauffman91] Kauffman, S. A. y Johnsen, S. Coevolution to the edge of chaos: Coupled fitness landscapes, poised states, and coevolutionary avalanches. *Journal of Theoretical Biology*, 149(4):467–505, 1991.

[Kaya11] Kaya, Y., Uyar, M., y Tekin, R. A novel crossover operator for genetic algorithms: Ring crossover. *CoRR*, abs/1105.0355, 2011.

[Kerschke14] Kerschke, P., Preuss, M., Hernández, C., Schütze, O., Sun, J.-Q., Grimme, C., Rudolph, G., Bischl, B., y Trautmann, H. Cell mapping techniques for exploratory landscape analysis. *En* A. Tantar, E. Tantar, J. Sun, W. Zhang, Q. Ding, O. Schtze, M. Emmerich, P. Legrand, M. P. Del, y C. C. Coello, eds., *EVOLVE — A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, tomo 288 de *Advances in Intelligent Systems and Computing*, págs. 115–131. Springer International Publishing, 2014. ISBN 978-3-319-07493-1. Publication status: Published.

[Kimura83] Kimura, M. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, Cambridge, 1983.

[Knjazew12] Knjazew, D. *OmeGA: A competent genetic algorithm for solving permutation and scheduling problems*, tomo 6. Springer Science & Business Media, 2012.

[Lipsich91] Lipsich, M. Adaptation on rugged landscapes generated by iterated local interactions of neighboring genes. *En* R. K. Belew y L. B. Booker, eds., *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, págs. 128–135. Morgan Kaufmann, San Mateo, CA, 1991.

[Lobo04] Lobo, J., Miller, J. H., y Fontana, W. Neutrality in technological landscapes. Inf. téc., working paper, Santa Fe Institute, Santa Fe, 2004.

[Lu11] Lu, G., Li, J., y Yao, X. *Evolutionary Computation in Combinatorial Optimization: 11th European Conference, EvoCOP 2011, Torino, Italy, April 27-29, 2011. Proceedings*, cap. Fitness-Probability Cloud and a Measure of Problem Hardness for Evolutionary Algorithms, págs. 108–117. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-20364-0.

[Magalhaes-Mendes13] Magalhaes-Mendes, J. A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS Trans. Comput*, 12(4):164–173, 2013.

[Malan09] Malan, K. y Engelbrecht, A. P. Quantifying ruggedness of continuous landscapes using entropy. *En IEEE Congress on Evolutionary Computation*, págs. 1440–1447. IEEE, 2009.

[Malan13] Malan, K. M. y Engelbrecht, A. P. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241(0):148 – 163, 2013. ISSN 0020-0255.

[Malan14] Malan, K. y Engelbrecht, A. Particle swarm optimisation failure prediction based on fitness landscape characteristics. *En Swarm Intelligence (SIS), 2014 IEEE Symposium on*, págs. 1–9. Dec 2014.

[Mattfeld99] Mattfeld, D. C. y Bierwirth, C. A search space analysis of the job shop scheduling problem. *Annals of Operations Research*, 86:441–453, 1999.

[Merkurjeva11] Merkurjeva, G. y Bolshakovs, V. Benchmark fitness landscape analysis. *International Journal of Simulation Systems, Science and Technology*, 12(2):38–45, 2011.

[Mersmann11] Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., y Rudolph, G. Exploratory landscape analysis. *En Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, págs. 829–836. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0557-0.

[Mersmann13] Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J., y Neumann, F. A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, 69(2):151–182, 2013. ISSN 1573-7470.

[Mitchell96] Mitchell, M. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[Müller11] Müller, C. L. y Sbalzarini, I. F. Global characterization of the CEC 2005 fitness landscapes using fitness-distance analysis. *En* C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekart, A. I. Esparcia-Alcazar, J. J. Merelo, F. Neri, M. Preuss, H. Richter, J. Togelius, y G. N. Yannakakis, eds., *Applications of Evolutionary Computing, EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, EvoSTOC*, tomo 6624 de *LNCS*, págs. 291–300. Springer Verlag, Turin, Italy, abr. 2011.

[Muñoz15] Muñoz, M. A., Kirley, M., y Halgamuge, S. K. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE Trans. Evolutionary Computation*, 19(1):74–87, 2015.

[Naudts00a] Naudts, B. y Kallel, L. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15, 2000.

[Naudts00b] Naudts, B. y Kallel, L. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 4(1):1–15, Apr 2000. ISSN 1089-778X.

[Nocedal06] Nocedal, J. y Wright, S. J. *Numerical Optimization, second edition.* World Scientific, 2006.

[Ochoa08] Ochoa, G., Tomassini, M., Verel, S., y Darabos, C. A study of nk landscapes' basins and local optima networks. *En* C. Ryan y M. Keijzer, eds., *GECCO*, págs. 555–562. ACM, 2008. ISBN 978-1-60558-130-9.

[Ortiz-Boyer07] Ortiz-Boyer, D., Hervás-Martínez, C., y García-Pedrajas, N. Improving crossover operator for real-coded genetic algorithms using virtual parents. *J. Heuristics*, 13(3):265–314, 2007.

[Picek09] Picek, S. y Golub, M. The new negative slope coefficient measure. *En Proceedings of the 10th WSEAS International Conference on Evolutionary Computing*, EC'09, págs. 96–101. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2009. ISBN 978-960-474-067-3.

[Pitzer10] Pitzer, E., Affenzeller, M., y Beham, A. A closer look down the basins of attraction. *En Computational Intelligence (UKCI), 2010 UK Workshop on*, págs. 1–6. Sept 2010.

[Pitzer12] Pitzer, E. y Affenzeller, M. A comprehensive survey on fitness landscape analysis. *En* J. C. Fodor, R. Klempous, y C. P. S. Araujo, eds., *Recent Advances in Intelligent Engineering Systems*, tomo 378 de *Studies in Computational Intelligence*, págs. 161–191. Springer, 2012. ISBN 978-3-642-23228-2.

[Reeves95] Reeves, C. R. y Wright, C. C. Epistasis in genetic algorithms: An

experimental design perspective. *En Proc. of the 6th International Conference on Genetic Algorithms, (pp 217–224*, págs. 217–224. Morgan Kaufmann, 1995.

[Reidys02] Reidys, C. M. y Stadler, P. F. Combinatorial landscapes. *SIAM Rev.*, 44(1):3–54 (electron, 2002. ISSN 0036-1445.

[Rice76] Rice, J. R. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[Rodriguez-Maya14a] Rodriguez-Maya, N. E., Graff, M., y Flores, J. J. *Nature-Inspired Computation and Machine Learning: 13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutiérrez, Mexico, November 16-22, 2014. Proceedings, Part II*, cap. Performance Classification of Genetic Algorithms on Continuous Optimization Problems, págs. 1–12. Springer International Publishing, Cham, 2014. ISBN 978-3-319-13650-9.

[Rodriguez-Maya14b] Rodriguez-Maya, N. E., Martínez-Carranza, J., Flores, J. J., y Graff, M. Solving a scholar timetabling problem using a genetic algorithm - study case: Instituto tecnologico de zitacuaro. *En 13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutierrez, Mexico, November 16-22, 2014, Special Session Proceedings*, págs. 197–202. 2014.

[Smith02] Smith, T., Philippides, A., Husbands, P., y O'Shea, M. Neutrality and ruggedness in robot landscapes. *En Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, tomo 2, págs. 1348–1353. 2002.

[Spears91] Spears, W. M. y Anand, V. A study of crossover operators in genetic programming. *En* Z. W. Ras y M. Zemankova, eds., *Proceedings of the Sixth International Symposium on Methodologies for Intelligent*

Systems ISMIS 91, tomo 542 de *Lecture Notes in Computer Science*, págs. 409–418. Springer-Verlag, October 16-19 1991. ISBN 3-540-54563-8.

[Stanhope98] Stanhope, S. A. y Daida, J. M. *Evolutionary Programming VII: 7th International Conference, EP98 San Diego, California, USA, March 25–27, 1998 Proceedings*, cap. Optimal mutation and crossover rates for a genetic algorithm operating in a dynamic environment, págs. 693–702. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-68515-9.

[Trujillo11] Trujillo, L., Martínez, Y., López, E. G., y Legrand, P. Predicting problem difficulty for genetic programming applied to data classification. *En GECCO*, págs. 1355–1362. 2011.

[Trujillo12a] Trujillo, L., Martinez, Y., Galvan-Lopez, E., y Legrand, P. A comparison of predictive measures of problem difficulty for classification with Genetic Programming. *En ERA 2012*. Tijuana, Mexico, nov. 2012.

[Trujillo12b] Trujillo, L., Martínez, Y., López-Galván, E., y Legrand, P. A comparative study of an evolvability indicator and a predictor of expected performance for genetic programming. *En Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*, GECCO Companion '12, págs. 1489–1490. ACM, New York, NY, USA, 2012. ISBN 978-1-4503-1178-6.

[Vanneschi02] Vanneschi, L. y Tomassini, M. A study on fitness distance correlation and problem difficulty for genetic programming. *En* S. Luke, C. Ryan, y U.-M. O'Reilly, eds., *Graduate Student Workshop*, págs. 307–310. AAAI, New York, 8 July 2002.

[Vanneschi04a] Vanneschi, L., Clergue, M., Collard, P., Tomassini, M., y Verel, S. Fitness clouds and problem hardness in genetic programming. *En* K. Deb, ed., *Genetic and Evolutionary Computation GECCO 2004*, tomo 3103 de *Lecture Notes in Computer Science*, págs. 690–701. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22343-6.

[Vanneschi04b] Vanneschi, L., Clergue, M., Collard, P., Tomassini, M., y Verel, S. Fitness clouds and problem hardness in genetic programming. *En* K. Deb, ed., *Genetic and Evolutionary Computation - GECCO 2004*, tomo 3103 de *Lecture Notes in Computer Science*, págs. 690–701. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22343-6.

[Vanneschi05] Vanneschi, L., Tomassini, M., Collard, P., y Clergue, M. A survey of problem difficulty in genetic programming. *En* S. Bandini y S. Manzoni, eds., *AI*IA 2005: Advances in Artificial Intelligence*, tomo 3673 de *Lecture Notes in Computer Science*, págs. 66–77. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-29041-4.

[Vanneschi06a] Vanneschi, L., Tomassini, M., Collard, P., y Vérel, S. Negative slope coefficient: A measure to characterize genetic programming fitness landscapes. *En Genetic Programming, 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006, Proceedings*, págs. 178–189. Springer, 2006.

[Vanneschi06b] Vanneschi, L., Tomassini, M., Collard, P., y Verel, S. Negative slope coefficient: A measure to characterize genetic programming fitness landscapes. *En Genetic Programming*, págs. 178–189. Springer, 2006.

[Vanneschi06c] Vanneschi, L., Tomassini, M., Pirola, Y., Vérel, S., y Mauri, G. A quantitative study of neutrality in gp boolean landscapes. *En Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'06*, págs. 895–902. ACM Press, 2006.

[Vanneschi08] Vanneschi, L. Investigating problem hardness of real life applications. *En* R. Riolo, T. Soule, y B. Worzel, eds., *Genetic Programming Theory and Practice V*, Genetic and Evolutionary Computation Series, págs. 107–124. Springer US, 2008. ISBN 978-0-387-76307-1.

[Vanneschi09] Vanneschi, L., Verel, S., Tomassini, M., y Collard, P. Nk landscapes difficulty and negative slope coefficient: How sampling influences the results. *En* M. Giacobini, A. Brabazon, S. Cagnoni, G. Di Caro, A. Ekrt, A. Esparcia-Alczar, M. Farooq, A. Fink, y P. Machado, eds., *Applications of Evolutionary Computing*, tomo 5484 de *Lecture Notes in Computer Science*, págs. 645–654. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01128-3.

[Vasconcelos01] Vasconcelos, J. A., Ramirez, J. A., Takahashi, R. H. C., y Saldanha, R. R. Improvements in genetic algorithms. *Magnetics, IEEE Transactions on*, 37(5):3414–3417, 2001.

[Vassilev99] Vassilev, V. K., Miller, J. F., y Fogarty, T. C. Digital circuit evolution and fitness landscapes. *En* P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, y A. Zalzala, eds., *Proceedings of the Congress on Evolutionary Computation*, tomo 2. IEEE Press, Mayflower Hotel, Washington D.C., USA, 6-9 jul. 1999. ISBN 0-7803-5536-9 (softbound).

[Vassilev00] Vassilev, V. K., Fogarty, T. C., y Miller, J. F. Information characteristics and the structure of landscapes. *Evolutionary Computation*, 8(1):31–60, 2000.

[Verel03] Verel, S. y Clergue, M. Where are bottleneck in nk fitness landscapes. *En Gedeon (Eds.), Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, IEEE*, págs. 273–280. Press, 2003.

[Verel07] Verel, S., Collard, P., y Clergue, M. Where are bottlenecks in nk fitness landscapes? *CoRR*, abs/0707.0641, 2007.

[Wagner96] Wagner, G. P. y Altenberg, L. Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, 1996.

[Weinberger90] Weinberger, E. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336, 1990. ISSN 1432-0770.

[Weise09] Weise, T. *Global Optimization Algorithms – Theory and Application*. it-weise.de (self-published): Germany, 2009.

[Wolpert97] Wolpert, D. H. y Macready, W. G. No free lunch theorems for optimization. *Trans. Evol. Comp*, 1(1):67–82, abr. 1997. ISSN 1089-778X.

[Wright32] Wright, S. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1:356–66, 1932.

[Xin09] Xin, B., Chen, J., y Pan, F. Problem difficulty analysis for particle swarm optimization: Deception and modality. *En Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, GEC '09, págs. 623–630. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-326-6.

[Yoon12] Yoon, Y. y Kim, Y.-H. *The Roles of Crossover and Mutation in Real-Coded Genetic Algorithms*. INTECH Open Access Publisher, 2012.