On the Taxonomy of Optimization Problems Under Estimation of Distribution Algorithms

Carlos Echegoyen carlos.echegoyen@ehu.es
Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, The University of the Basque Country (UPV/EHU). Paseo Manuel de Lardizábal 1. 20018 San Sebastian, Spain

Alexander Mendiburu alexander.mendiburu@ehu.es
Intelligent Systems Group, Department of Computer Architecture and Technology, The University of the Basque Country (UPV/EHU). Paseo Manuel de Lardizábal 1. 20018 San Sebastian, Spain

Roberto Santana roberto.santana@ehu.es
Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, The University of the Basque Country (UPV/EHU). Paseo Manuel de Lardizábal 1. 20018 San Sebastian, Spain

Jose A. Lozano ja.lozano@ehu.es
Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, The University of the Basque Country (UPV/EHU). Paseo Manuel de Lardizábal 1. 20018 San Sebastian, Spain

doi:10.1162/EVCO_a_00095

Abstract

Understanding the relationship between a search algorithm and the space of problems is a fundamental issue in the optimization field. In this paper, we lay the foundations to elaborate taxonomies of problems under estimation of distribution algorithms (EDAs). By using an infinite population model and assuming that the selection operator is based on the rank of the solutions, we group optimization problems according to the behavior of the EDA. Throughout the definition of an equivalence relation between functions it is possible to partition the space of problems in equivalence classes in which the algorithm has the same behavior. We show that only the probabilistic model is able to generate different partitions of the set of possible problems and hence, it predetermines the number of different behaviors that the algorithm can exhibit. As a natural consequence of our definitions, all the objective functions are in the same equivalence class when the algorithm does not impose restrictions to the probabilistic model. The taxonomy of problems, which is also valid for finite populations, is studied in depth for a simple EDA that considers independence among the variables of the problem. We provide the sufficient and necessary condition to decide the equivalence between functions and then we develop the operators to describe and count the members of a class. In addition, we show the intrinsic relation between univariate EDAs and the neighborhood system induced by the Hamming distance by proving that all the functions in the same class have the same number of local optima and that they are in the same ranking positions. Finally, we carry out numerical simulations in order to analyze the different behaviors that the algorithm can exhibit for the functions defined over the search space $\{0, 1\}^3$.

Keywords

Estimation of distribution algorithms, probabilistic models, factorizations, rank-based selection, model of infinite population, equivalence classes, taxonomy of functions, neighborhood system, dynamical systems.

Manuscript received: January 31, 2012; revised: August 16, 2012; accepted: October 24, 2012.

1 Introduction

Estimation of distribution algorithms (EDAs; Mühlenbein and Paaß, 1996; Larrañaga and Lozano, 2002; Pelikan, 2005) are a population-based optimization paradigm in the field of evolutionary computation that has acquired special relevance in the last decade. Proof of this popularity is the development of new and more complex EDAs (Bosman, 2010; Hauschild et al., 2012), the applications for these EDAs (Armañanzas et al., 2008; Santana et al., 2008; Brownlee et al., 2008) and the works which study fundamental issues in order to better understand how these algorithms perform (González et al., 2002; Zhang, 2004; Shapiro, 2005; Hauschild et al., 2009; Echegoyen et al., 2011).

The main characteristic of EDAs is the use of probabilistic models to lead the search toward promising areas of the space of solutions. By making use of a subset of promising solutions belonging to the population, the learning algorithm allows an estimate to be made of a new probability distribution over the search space at each step of the algorithm. Thus, each of the possible solutions has an associated probability of being sampled, which varies during the optimization process. The probability values assigned to the solutions are the main source of information in determining which solutions will be returned by the algorithm. Consequently, given a problem, the ideal objective of an EDA is to get higher probability values for the highest quality solutions throughout an iterative process. However, when an EDA is applied to search for the best solutions of a problem, the algorithm can exhibit a wide variety of behaviors (Echegoyen et al., 2011). In this regard, identifying and classifying the different behaviors of an EDA, and how these behaviors relate to the characteristics of the optimization problems, are fundamental issues to gain an understanding of the underlying mechanisms that govern this type of algorithm.

The behavior of an EDA can be specified by the sequence of probability distributions generated at each generation. Ultimately, the algorithm can have different executions depending on the objective function. However, with the aim of developing a taxonomy of problems, we start by considering the following question: Is it possible to group the optimization problems according to the behavior of the EDA? Therefore, we are questioning the existence of groups of problems in which the algorithm exhibits a similar performance regarding the optimization purpose. In order to start the study of this topic, we make the following three assumptions: (i) we consider EDAs with infinite population, (ii) the selection scheme is based on the rank of the solutions, and (iii) the algorithm is applied in the space of injective functions.

One important particularity of the approach carried out in the current paper is the use of ranking information in the selection step to guide the search. We argue that many black box algorithms only compare objective values instead of taking into account their absolute values. In addition, ranking-based algorithms play an important role in the theory of black box optimization. For instance, it has been shown in Doerr and Winzen (2012) that, for certain optimization problems, the use of ranking-based selection mechanisms facilitates the estimates of realistic time complexity results.

In order to elaborate taxonomies of optimization problems under EDAs that satisfy the aforementioned three assumptions, a crucial element is the definition of an equivalence relation between objective functions. The equivalence relation, which is based on the sequences of probability distributions generated during the search, partitions the space of functions into equivalence classes. We will see that it is possible to assert that the algorithm has the same behavior for the functions belonging to the same class. In turn, we show that only the probabilistic model is able to create partitions of the space of problems. From the definitions that we provide, it can be deduced that all

the objective functions are equivalent when the probabilistic model is able to exactly recover the underlying distribution of the selected individuals. Therefore, we can say that all the optimization problems are equally difficult from the point of view of this type of EDA. This crucial role of the probabilistic model supports the importance that has been attributed to it in the literature regarding the classification of different EDAs and the development of new techniques and studies.

The partition of the space of problems under EDAs is studied in depth for a so-called univariate algorithm whose probabilistic model assumes independence among the variables of the problem. The key point to understanding the behavior of an EDA for a given problem is the way in which the objective function is related to the probabilistic model. We show that this relation can be expressed by means of a structure of sets. For this type of univariate EDA, the necessary and sufficient condition to identify equivalent functions is provided. Next, we develop the operators that allow us to describe the functions in the same class and count the number of elements per class. Another fundamental topic that we take into account is the relation between the behavior of the EDA and the properties of the objective function. In particular, we study the connection between the equivalence classes and the local optima of the objective functions belonging to each class. We show that the functions in the same class have the same number of local optima and in the same ranking positions. This fact reveals the intrinsic connection between local optima and any EDA that introduces a univariate probabilistic model. This link has only been shown for specific EDA implementations (González et al., 2001; Zhang, 2004).

Finally, we conduct numerical simulations of a univariate EDA that implements tournament selection (Zhang, 2004). The algorithm is applied to the injective functions defined over the search space $\{0, 1\}^3$. The partition of the space of problems in equivalence classes allows us to carry out a detailed analysis of the different EDA behaviors. Through the numerical analysis, we mainly study the complexity of the problems belonging to each class. Due to the relevant role that the local optima play in EDAs that assume independence among the variables, we present the difficulty of the problems in relation to that number. Moreover, the experiments are useful in order to illustrate the taxonomy of problems which is abstractly presented throughout the paper.

The rest of the paper is organized as follows. Section 2 formally introduces the problems to solve and the estimation of distribution algorithms. Section 3 presents EDAs with infinite population and provides the specific definitions involved in the framework of the current work. In Section 4, the concept of equivalence between functions is presented and discussed. Section 5 regards the study of the equivalence classes under a simple EDA. Firstly, it explains and establishes the sufficient and necessary condition to decide the equivalence between two functions. Secondly, the operators to describe the functions in the same class are deduced. Lastly, the relationship between the equivalence classes and local optima is presented. In Section 6, numerical experiments are conducted. Section 7 points out possible future work. Finally, Section 8 draws the conclusions obtained during the study.

2 Background

2.1 Optimization Problems

In this paper we consider the following general optimization problem:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x}), \tag{1}$$

Table 1: Example of permutation σ induced by an injective function with $n = 3$ variables.

$f(\mathbf{x})$	σ	Rank
100	(1, 1, 1)	1
50	(0, 1, 0)	2
45	(0, 0, 1)	3
20	(1, 0, 0)	4
10	(0, 1, 1)	5
3	(1, 0, 1)	6
1	(1, 1, 0)	7
0	(0, 0, 0)	8

where $S = \{0, 1\}^n$ is the search space, $\mathbf{x} = (x_1, \ldots, x_n) \in S$ is a solution and $f : S \to \mathbb{R}$ is the objective function. For the sake of simplicity, we deal with binary solutions although the fundamental results provided do not depend on the cardinality of the variables or the codification of the solutions. The cardinality $|S|$ of the search space is therefore 2^n. Throughout the paper, we assume that the function $f(\mathbf{x})$ is injective. Thus, for each $z \in f(S)$, there is only one $\mathbf{x} = (x_1, \ldots, x_n) \in S$ such that $f(\mathbf{x}) = z$. The results could be generalized to noninjective functions because the most basic definitions that we present do not depend on the injectivity of the function.

A crucial fact that we use in the development of the current work is the following. A function $f(\mathbf{x})$ naturally induces a permutation σ of the solutions of the search space S. This permutation σ is thought of as a 2^n-tuple of the elements in S ordered according to function values $f(S)$. The first solution of σ, namely $\mathbf{x}_1$, has the highest function value, $\mathbf{x}_2$ has the second highest, and so on, with the last solution $\mathbf{x}_{2^n}$ being the one with the lowest function value:

$$f(\mathbf{x}_1) > f(\mathbf{x}_2) > \cdots > f(\mathbf{x}_{2^n}).$$

Of course, the first solution $\mathbf{x}_1$ of σ corresponds to the solution $\mathbf{x}^*$ that solves Equation (1). Independent of the specific function values $f(S)$, whenever they provide the same ranking of the solutions $\mathbf{x} \in S$, the function is represented by the same permutation σ. In the case of injective functions, we have $2^n!$ permutations σ and the set containing all possible σ is denoted by Π. A σ permutation provides a ranking of the solutions of the search space. From now on, the objective function $f(\mathbf{x})$ and the corresponding σ can be used indistinctly as synonyms. We use the notation $\sigma[i]$ to index the permutation. Thus, $\sigma[i] = \mathbf{x}$ returns the solution $\mathbf{x} \in S$ which is at position i in the ranking.

In Table 1, we provide an example of the permutation σ induced by a function $f(\mathbf{x})$. In the first column, the original function values are shown. In the second column, the corresponding 2^n-tuple σ is presented; and in the third column, we indicate the rank of the solutions.

2.2 Estimation of Distribution Algorithms

In the field of evolutionary computation, estimation of distribution algorithms (EDAs) arise, in part, as an alternative to genetic algorithms (GAs; Goldberg, 1989). Instead of exchanging information between individuals through genetic operators, EDAs use machine learning methods (Larrañaga and Lozano, 2002) to extract relevant features of the search space through the selected individuals of the population. The collected

Algorithm 1 EDA scheme

1: $D_{t=0} \leftarrow$ Generate N individuals randomly
2: **do** $\{$
3: $D_t \leftarrow$ Evaluate individuals
4: $D_t^s \leftarrow$ Select $M < N$ individuals from D_t according to a selection method
5: $p_t(\boldsymbol{x}) = p(\boldsymbol{x}|D_t^s) \leftarrow$ Estimate the joint probability distribution by means of a probabilistic model
6: $D_{t+1} \leftarrow$ Sample M individuals from $p_t(\boldsymbol{x})$ and create the new population
7: $t \leftarrow t + 1$
8: $\}$ **until** Stopping criterion is met

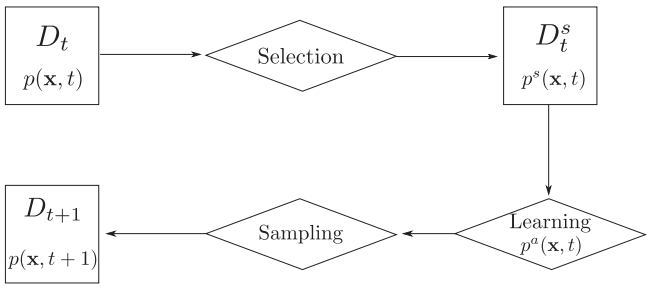


Figure 1: Probability distributions determined by the components of an EDA. D_t, D_t^s, D_{t+1}: population and selected individuals at generation t and population at generation $t + 1$; $p(\mathbf{x}, t)$, $p^s(\mathbf{x}, t)$, $p^a(\mathbf{x}, t)$: Probability distributions of the population, the selected set and the probabilistic model approximation at generation t.

information is encoded by using a probabilistic model which is later employed to generate new solutions. Thus, EDAs use explicit probability distributions with the aim of finding the optimal solution $\mathbf{x}^*$ and solving Equation (1). Therefore, we need the following definitions. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a vector of n binary random variables. We define $p(\mathbf{X} = \mathbf{x})$, or simply $p(\mathbf{x})$, as the joint probability distribution of the variables $\mathbf{X}$ taking values from the search space S. The general scheme of the EDA approach is shown in Algorithm 1.

The procedure of an EDA can be explained in terms of the probability distributions involved in the search (Santana et al., 2009; see Figure 1). Firstly, D_t denotes the EDA population at generation t and $p(\mathbf{x}, t)$ is the underlying probability distribution of this sample. Secondly, $p^s(\mathbf{x}, t)$ is the probability distribution of the selected individuals D_t^s. Finally, $p^a(\mathbf{x}, t)$ is the distribution given by the probabilistic model chosen to approximate $p^s(\mathbf{x}, t)$. Once we have $p^a(\mathbf{x}, t)$, the next generation D_{t+1} is constructed by sampling this distribution.

In any evolutionary algorithm (EA), selection is one of the fundamental operators and therefore a wide variety of proposals can be found in the literature. According to Lee and El-Sharkawi (2008), the selection schemes can be classified into two groups: proportionate selection and ordinal-based selection. Proportionate-based procedures select individuals based on their specific function values. Ordinal-based selection

procedures select individuals based on their rank within the population. Thus, this class of selection only takes into account qualitative information about the function, that is, it only uses the fact that $f(\mathbf{x}) > f(\mathbf{y})$ instead of the real value given by the function. In this work, we assume that the algorithm uses this last type of selection. We must say that assuming ordinal-based selection is not an insurmountable restriction, because the definitions that we present could be extended to selections that take into account the specific function values. Common examples of selection schemes based only on the rank of the solutions are truncation, tournament, linear ranking, and exponential ranking selection. These schemes are considered in a wide variety of EAs, both in solving real problems and in theoretical studies (Blickle and Thiele, 1996; Prügel-Bennett, 2000; Zhang, 2004; Doerr and Winzen, 2012).

3 The Infinite Population EDA Model

The application of the EDA scheme presented in Algorithm 1 to face optimization problems can involve an unapproachable variety of situations and behaviors. With the aim of dealing with all possible EDA behaviors, we consider the concept of infinite population (Vose, 1999), which is commonly used to analyze EAs. Regarding the field of EDAs, several works have assumed an infinite population in order to study and provide the fundamental properties of this type of algorithm (e.g., Mühlenbein et al., 1999; Zhang and Mühlenbein, 2004; Zhang, 2004).

In EDAs with infinite population, it is assumed that the empirical probability distribution induced by the solutions in D_t and D_t^s (see Figure 1) will converge to the underlying probability distributions $p(\mathbf{x}, t)$ and $p^s(\mathbf{x}, t)$, respectively, as the size of the sample tends to infinity. Therefore, $p(\mathbf{x}, t)$ and $p^s(\mathbf{x}, t)$ could be thought of as the population and the selected individuals at iteration t (Zhang and Mühlenbein, 2004). Consequently, we can assume that $p(\mathbf{x}, t+1) = p^a(\mathbf{x}, t)$. In EDAs with infinite population, the random errors caused by sampling finite sets of solutions from the probability distributions managed by the algorithm are canceled and hence, we no longer need D_t and D_t^s.

By assuming infinite population, the EDA is seen as a sequence of probability distributions. Therefore, we concentrate solely on the evolution of these probability distributions during the search instead of dealing with the specific candidate solutions. Nevertheless, it is important to note that, although the analysis is carried out by assuming infinite populations, the taxonomy of problems developed throughout the paper is also valid for finite populations.

3.1 The Algorithm and the Population

In Algorithm 2 we formally describe the infinite population EDA procedure that we consider in the current work. This algorithm will be simply denoted by $\mathcal{A}$.

It is established that the first step in algorithm $\mathcal{A}$ is to create the permutation σ of the solutions $\mathbf{x} \in S$. The permutation σ that the objective function provides is a crucial element of the algorithm.

As commented above, an infinite population can be represented by the probability distribution $p(\mathbf{x}, t)$ that the algorithm manages at time t. Nevertheless, algorithm $\mathcal{A}$ does not explicitly work with the distributions $p(\mathbf{x}, t)$. Instead, at each generation, this algorithm manages a probability vector $\mathbf{p} = (p_1, p_2, \ldots, p_{|S|})$ where each probability value p_i in $\mathbf{p}$ corresponds to the probability of the solution $\mathbf{x}_i$, which is in the position i of the permutation σ. It is always interpreted that the first value p_1 of the vector $\mathbf{p}$ is the probability of the optimal solution, p_2 is the probability of the second best solution, and so on, with the last probability p_{2^n} corresponding to the solution with the worst

Algorithm 2 EDA with infinite population, $\mathcal{A}$

$\sigma \leftarrow$ permutation of the solutions $\mathbf{x} \in S$ given by $f(\mathbf{x})$
$\mathbf{p}_{t=0} \leftarrow$ generate the initial population
do $\{$
    Compute the probability of selection through the selection operator ϕ as $\mathbf{p}_t^s = \phi(\mathbf{p}_t)$
    Compute $\mathbf{p}_t^a = \mathcal{M}(\mathbf{p}_t^s, \sigma)$ to approximate $\mathbf{p}_t^s$.
    $\mathbf{p}_{t+1} \leftarrow \mathbf{p}_t^a$
    $t \leftarrow t + 1$
$\}$ **until** Convergence

function value. Accordingly, we have the probability vectors $\mathbf{p}^s$ and $\mathbf{p}^a$ representing the selected population and the approximation respectively. To be absolutely precise, the probability vectors at time t should be represented as $\mathbf{p}_t = (p_1^t, p_2^t, \ldots, p_{|S|}^t)$. Note that, unlike the probability distribution $p(\mathbf{x})$, a vector $\mathbf{p}$ implies no specific assignment of probabilities to solutions $\mathbf{x}$. In order to obtain the probability distribution $p(\mathbf{x}, t)$ that algorithm $\mathcal{A}$ manages at time t, we link the vector $\mathbf{p}_t$ and the permutation σ through the pair $(\mathbf{p}, \sigma)$. The pair $(\mathbf{p}, \sigma)$ induces a probability function $p(\mathbf{x})$ such that $p(\sigma[i]) = p_i$. Therefore, we have that $p_1 = p(\sigma[1])$ is the probability of the optimum, $p_2 = p(\sigma[2])$ is the probability of the second best solution, and so on.

The arrangement of the probability vector $\mathbf{p}$ and its relationship with the permutation σ is illustrated in Table 2. In this example, we consider two different permutations σ_1 and σ_2. We assume that algorithm $\mathcal{A}$ was applied to both functions and that it manages the same probability vector $\mathbf{p}_t$ at time t. We can see that, for example, p_1 is associated with the solution $(1, 1, 1)$ according to σ_1. However, according to σ_2, p_1 is associated with the solution $(0, 0, 0)$. Therefore, p_1 is the probability of the optimum in both cases, independent of the specific configuration of this solution. In the third column of Table 2, we can see that the same vector induces different probability distributions depending on σ. Note that, since we have $2^n!$ different permutations σ, a vector $\mathbf{p}$ can generate $2^n!$ different probability distributions.

The space of the possible probability vectors that the algorithm can generate is defined by the following simplex:

$$\Omega_{|S|} = \{(p_1, p_2, \ldots, p_{|S|}) : \sum_{i=1}^{|S|} p_i = 1,\ p_i \geq 0\}.$$

In addition, we establish that any initial vector $\mathbf{p}_0 = (p_1, \ldots, p_{2^n})$ satisfies that $p_i < 1$ for all $i \in \{1, \ldots, 2^n\}$. This condition is established in order to avoid the trivial case in which the algorithm starts from a degenerate probability distribution that assigns 1 to a solution of the search space. In this case, the algorithm would converge when the initial population is generated. This situation will not be considered.

From any given $\mathbf{p}_0$, the application of algorithm $\mathcal{A}$ to a function f induces a deterministic sequence of probability vectors:

$$\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots. \tag{2}$$

We use this sequence to describe the behavior of the algorithm. Note that if we took into account the probability distributions $p(\mathbf{x}, t)$ to describe the EDA behavior, the algorithm

Table 2: Example with $n = 3$ variable of the arrangement of the probability vectors and their relationship with the probability distributions.

σ_1	$\mathbf{p}_t$	$p(\mathbf{x}, t)$	$\mathbf{p}_t^s$
(1,1,1)	p_1	$p(\sigma[1]) = p(1,1,1)$	p_1^s
(0,1,0)	p_2	$p(\sigma[2]) = p(0,1,0)$	p_2^s
(0,0,1)	p_3	$p(\sigma[3]) = p(0,0,1)$	p_3^s
(1,0,0)	p_4	$p(\sigma[4]) = p(1,0,0)$	p_4^s
(0,1,1)	p_5	$p(\sigma[5]) = p(0,1,1)$	p_5^s
(1,0,1)	p_6	$p(\sigma[6]) = p(1,0,1)$	p_6^s
(1,1,0)	p_7	$p(\sigma[7]) = p(1,1,0)$	p_7^s
(0,0,0)	p_8	$p(\sigma[8]) = p(0,0,0)$	p_8^s

σ_2	$\mathbf{p}_t$	$p(\mathbf{x}, t)$	$\mathbf{p}_t^s$
(0,0,0)	p_1	$p(\sigma[1]) = p(0,0,0)$	p_1^s
(1,0,1)	p_2	$p(\sigma[2]) = p(1,0,1)$	p_2^s
(1,1,0)	p_3	$p(\sigma[3]) = p(1,1,0)$	p_3^s
(0,1,1)	p_4	$p(\sigma[4]) = p(0,1,1)$	p_4^s
(1,0,0)	p_5	$p(\sigma[5]) = p(1,0,0)$	p_5^s
(0,1,0)	p_6	$p(\sigma[6]) = p(0,1,0)$	p_6^s
(0,0,1)	p_7	$p(\sigma[7]) = p(0,0,1)$	p_7^s
(1,1,1)	p_8	$p(\sigma[8]) = p(1,1,1)$	p_8^s

could always generate different sequences of distributions for each possible σ. The use of probability vectors $\mathbf{p}$ provides a higher level of abstraction, which is essential in order to group EDA behaviors.

3.2 The Selection Scheme ϕ

As indicated previously, we assume selection schemes based on the rank of the solutions within the population. Since in algorithm $\mathcal{A}$ any probability value p_i is interpreted as the probability of the solution $\sigma[i]$ with position i in the permutation σ, the selection can be simply defined by a function $\phi : \Omega_{|S|} \to \Omega_{|S|}$. This function modifies the probability values of the individuals according to the rank in which they are.

In order to create the partition of the space of problems, we do not need to consider any specific implementation of the selection operator ϕ. However, it will be essential for ϕ to satisfy two basic properties or axioms.

- **Property 1 (Impartiality).** The selection operator is independent of the configuration $\mathbf{x}$ of a solution. This operator only takes into account the fitness value $f(\mathbf{x})$ of the solution. Although this property is implicit in the definition of ϕ, we believe that it is worth a brief discussion. Thus, since we assume an ordinal-based selection scheme, given $\mathbf{p}$, we always obtain the same the probability vector $\mathbf{p}^s = \phi(\mathbf{p})$ after selection, independent of σ. This fact is illustrated in the last column of Table 2 where we indicate that the probability vector after selection is the same in both functions.

- **Property 2 (No degeneration).** The selection operator cannot assign extreme probabilities 1 or 0 to solutions whose probabilities are in the interval $(0, 1)$. More formally, the vector $\mathbf{p}^s = (p_1^s, \ldots, p_{2^n}^s)$, computed as $\mathbf{p}_t^s = \phi(\mathbf{p}_t)$ at generation t, satisfies that if $0 < p_i < 1$ then $0 < p_i^s < 1$ for all $i \in \{1, \ldots, 2^n\}$ in every generation $t = 1, 2, 3, \ldots$. In addition, if $p_i = 0$ then $p_i^s = 0$. The convergence of the algorithm can only take place as a result of the iterative process when t tends to infinity.

The conditions mentioned above are needed in order to guarantee that the taxonomies of problems are valid for any implementation of ϕ. Therefore, the selection ϕ will play no role in the partition of the space of optimization problems.

In the context of EDAs, some examples of selection schemes implemented and studied under infinite population can be found in Mühlenbein et al. (1999), Zhang and Mühlenbein (2004), and Zhang (2004).

3.3 The Approximation Step $\mathcal{M}$

In algorithm $\mathcal{A}$, the approximation step deals with the probability distribution $p^s(\mathbf{x})$ of the selected individuals. Therefore, the probability vector $\mathbf{p}^s$ has to be related to the corresponding solutions $\mathbf{x}$ by means of the pair $(\mathbf{p}^s, \sigma)$. The approximation step is defined as a function $\mathcal{M} : \Omega_{|S|} \times \Pi \to \Omega_{|S|}$ and it is computed as $\mathbf{p}^a = \mathcal{M}(\mathbf{p}^s, \sigma)$ in the algorithm.

Note that the approximation $\mathcal{M}$ is the only operator in $\mathcal{A}$ that takes into account the permutation σ. Therefore, $\mathcal{M}$ can translate the difference between functions to different behaviors of the algorithm.

4 Equivalent Functions and Equivalence Classes

In this section, we discuss the concept of equivalence between functions and provide the formal definitions. An EDA $\mathcal{A}$ induces deterministic sequences of probability vectors. An iteration of this algorithm is computed by a function $\mathcal{G} : \Omega_{|S|} \times \Pi \to \Omega_{|S|}$ as $\mathbf{p}_{t+1} = \mathcal{G}(\mathbf{p}_t, \sigma)$. The function $\mathcal{G}$ is a composition of the selection ϕ and the factorization function $\mathcal{M}$ (steps 4 and 5 in Algorithm 2) such that $\mathcal{G} = \mathcal{M} \circ \phi$. Thus, the sequence of probability vectors induced by $\mathcal{A}$ can be expressed as the iterations of the function $\mathcal{G}$:

$$\mathbf{p}_0, \mathcal{G}(\mathbf{p}_0, \sigma), \mathcal{G}^2(\mathbf{p}_0, \sigma), \mathcal{G}^3(\mathbf{p}_0, \sigma), \ldots$$

where $\mathcal{G}^t(\mathbf{p}_0, \sigma)$ is the vector at iteration t.

The properties of functions similar to $\mathcal{G}^t$ have usually been studied by means of discrete dynamical systems. In González et al. (2001) and Zhang (2004), important insights and results about the convergence of some EDAs were provided using this approach. In the current paper, the definition of equivalence does not consider specific implementations for $\mathcal{M}$ or ϕ and hence, we do not have a formulation of $\mathcal{G}$ to study the dynamics of the algorithms. Although the iterations of $\mathcal{G}$ are modeled as a dynamical system, we will take a more general perspective to describe the behavior of EDAs.

The definition of equivalence between objective functions under EDAs can be expressed as follows.

DEFINITION 1: *Let σ_1 and σ_2 be the permutations induced by the objective functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ respectively. Let $\mathcal{A}$ be an EDA with any given ϕ and $\mathcal{M}$. We say that σ_1 and σ_2 are equivalent under $\mathcal{A}$, and by extension f_1 and f_2, if for any $\boldsymbol{p}_0 \in \Omega_{|S|}$, $\mathcal{G}^t(\boldsymbol{p}_0, \sigma_1) = \mathcal{G}^t(\boldsymbol{p}_0, \sigma_2)$ for all $t = 1, 2, 3, \ldots$.*

Table 3: Example with $n = 2$ variables of two equal sequences when $\mathcal{A}$ is applied to σ_1 and σ_2.

σ_1	$\mathbf{p}_0$	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	...	$\mathbf{p}_\infty$
(1,1)	0.25	0.4375	0.6836	0.8999	...	1
(0,1)	0.25	0.3125	0.2539	0.0962		0
(0,0)	0.25	0.1875	0.0586	0.0039		0
(1,0)	0.25	0.0625	0.0039	0.0000	...	0

σ_2	$\mathbf{p}_0$	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	...	$\mathbf{p}_\infty$
(1,0)	0.25	0.4375	0.6836	0.8999	...	1
(1,1)	0.25	0.3125	0.2539	0.0962		0
(0,1)	0.25	0.1875	0.0586	0.0039		0
(0,0)	0.25	0.0625	0.0039	0.0000	...	0

In a less formal way, we say that two functions are equivalent under $\mathcal{A}$ if the corresponding sequences of probability vectors induced by the algorithm are equal from any initial point. The equivalence between functions means that we have the same EDA behavior if we focus on the rank of the solutions instead of their specific $\mathbf{x}$ configurations. Therefore, if two functions are equivalent, we can say that the algorithm will have the same performance in terms of solving Equation (1). In Table 3, a very simple example with $n = 2$ variables illustrates the equality of vector sequences. Departing from the uniform distribution, algorithm $\mathcal{A}$ exactly induces the same sequence of vectors $\mathbf{p}_t$. If both sequences are equal from any initial $\mathbf{p}_0$ then σ_1 and σ_2 will be equivalent. Note that the sequences of probability distributions $p(\mathbf{x}, t)$ are different.

Definition 1 provides an equivalence relation because it is a reflexive, symmetric, and transitive relation between functions. Given this equivalence relation, for each σ, we have the equivalence class of σ, denoted by $[\sigma]$. The equivalence relation partitions the space of functions into equivalence classes under an algorithm $\mathcal{A}$. The sequences of probability vectors generated by the algorithm uniquely identify the functions in a class.

The equivalence between functions is defined under a given algorithm $\mathcal{A}$ which implements certain ϕ and $\mathcal{M}$. However, as we discussed in the previous section, both operators do not play the same role. If two functions are equivalent under $\mathcal{A}$, then both functions will be equivalent for any given ϕ implemented in $\mathcal{A}$ satisfying Properties 1 and 2. However, two functions that are equivalent for $\mathcal{M}$ could not be so for $\mathcal{M}'$, which implements a different probabilistic model. We can conclude that only the factorization used to approximate $p^s(\mathbf{x})$ can create different partitions of the space of problems. This partition is independent of the implementation of ϕ.

By taking into account all the aforementioned definitions of algorithm, function, and equivalence, we can deduce the following result assuming that ϕ satisfies the properties of impartiality and no degeneration.

THEOREM 1: *Let $\mathcal{A}$ be an EDA whose implementation of $\mathcal{M}$ satisfies $p^a(\boldsymbol{x}, t) = p^s(\boldsymbol{x}, t)$ for all $t = 1, 2, 3, \ldots$. All the objective functions are in the same equivalence class under $\mathcal{A}$.*

PROOF: The proof of the theorem is straightforward. Two functions σ_1 and σ_2 are equivalent if $\mathcal{A}$ generates the same sequence of probability vectors departing from any initial $\mathbf{p}_0$. Due to the impartiality of ϕ, we obtain the same vector $\mathbf{p}_0^s = \phi(\mathbf{p}_0)$ after selection for σ_1 and σ_2. Next, if $\mathcal{M}$ satisfies $p^a(\mathbf{x}) = p^s(\mathbf{x})$, then $\mathcal{M}(\mathbf{p}_0^s, \sigma_1) = \mathcal{M}(\mathbf{p}_0^s, \sigma_2)$, and therefore we have the same vector $\mathbf{p}_0^a$ for both functions. Since $\mathbf{p}_{t+1} = \mathbf{p}_t^a$, the algorithm obtains the same probability vector $\mathbf{p}_1$, and consequently it will obtain the same vectors $\mathbf{p}_t$ at any time $t = 2, 3 \ldots$. □

Therefore, if it is assumed that $p(\mathbf{x}, t+1) = p^s(\mathbf{x}, t)$, the algorithm has the same behavior for all the injective functions and hence the same properties of convergence to the optimum (Mühlenbein et al., 1999; Zhang and Mühlenbein, 2004). As commented above, only the probabilistic model is able to create partitions of the space of problems. And moreover, if this model is able to exactly represent the distribution of the selected individuals, all the functions are in a single class. These results support the usual classification of EDAs which is carried out according to the probabilistic model implemented.

4.1 Descriptors of the Behavior of EDAs

Once the space of problems is divided into equivalence classes, we could study the behavior that the algorithm exhibits in each class. In this regard, we discuss the role of two descriptors of the behavior of the EDA which are relevant for the purposes of the current paper. On the one hand, the most basic descriptor is the sequences of probability vectors. We know that the algorithm induces the same set of sequences for any function in the same class and different sets of sequences for functions in different classes. Therefore, the set of sequences associated with any function of a class can be used to unequivocally represent the behavior of the corresponding class.

On the other hand, the behavior of an EDA can be described by using the basins of attraction of the solutions in S. Basin of attraction is a term used in dynamical systems that we adopt in a simplified manner. Roughly speaking, the basin of attraction of a point $\mathbf{x}$ in a dynamical system is the set of initial points that converge to $\mathbf{x}$. More formally, by using the function $\mathcal{G}$, we say that the basin of attraction of a solution $\mathbf{x}_i \in S$, is the set $\mathcal{Z} \subseteq \Omega_{|S|}$ such that $\forall \mathbf{p} \in \mathcal{Z}$, $\lim_{t\to\infty} \mathcal{G}^t(\mathbf{p}, \sigma) = (0, \ldots, p_i = 1, \ldots, 0)$ where $i \in \{1, \ldots, 2^n\}$ is the position of the solution $\mathbf{x}_i$ in the ranking σ. According to this definition, the basins of attraction related to each solution generate a partition of $\Omega_{|S|}$ that can be expressed by the sets $\mathcal{Z}_1, \ldots, \mathcal{Z}_{|S|}$.

The sequences of probability vectors express the complete process of convergence, whereas the basins of attraction represent only the final convergence of the algorithm. From Definition 1, we can deduce that if functions σ_1 and σ_2 are equivalent, the algorithm generates the same basins of attraction $\mathcal{Z}_1, \ldots, \mathcal{Z}_{|S|}$ for both functions. Therefore, all the functions belonging to the same class have the same basins of attraction. However, two different classes could have the same basins of attraction although they always have different sequences of probability vectors. Informally speaking, we might say that, for functions in different classes, the algorithm can reach the same places although it uses different roads.

5 Case Study: Univariate EDA

The concepts and definitions about the taxonomy of problems that were abstractly discussed in the previous sections will be studied in depth for a simple EDA that assumes independence among the variables of the problem. Different algorithms such

as population-based incremental learning (PBIL), compact genetic algorithm (cGA), and univariate marginal distribution algorithm (UMDA) introduce this type of model (Larrañaga and Lozano, 2002).

We consider that the function $\mathcal{M}$ approximates the distribution $p^s(\mathbf{x})$ by means of the following univariate factorization:

$$p^a(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{n} p^s(x_i|\boldsymbol{\theta}_i), \tag{3}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ is the set of parameters associated with the factorization. The local parameters $\boldsymbol{\theta}_i = (\theta_i^0, \theta_i^1)$ specify the marginal probability distributions $p^s(x_i)$ where $\theta_i^{x_i}$ is the probability value $p^s(X_i = x_i)$. In EDAs with finite population, each parameter $\theta_i^{x_i}$ can be estimated from the selected individuals by using different approaches such as relative frequencies or more sophisticated update rules. Nevertheless, in the model of infinite population, the marginal probabilities can be exactly calculated as,

$$p^s(x_i) = \sum_{\mathbf{x}\setminus x_i} p^s(\mathbf{x}). \tag{4}$$

From now on, we will ignore the explicit reference to the parameters $\boldsymbol{\theta}$ in Equation (3). Specifically, the function $\mathcal{M}$ is implemented in algorithm $\mathcal{A}$ as

$$p^a(\mathbf{x}) = \prod_{i=1}^{n} \sum_{\mathbf{x}\setminus x_i} p^s(\mathbf{x}). \tag{5}$$

5.1 Equivalence Condition

The key to understanding the relation between a given optimization problem and an EDA is the way in which the permutation σ is related to the probabilistic model used to compute the approximation step $\mathcal{M}$. More specifically, we take into account how the ranking positions τ of the solutions are organized in the different marginal distributions that form the factorization. These positions τ are grouped by using different sets as described below. The calculation of each marginal probability $p^s(x_i)$ can be put into relation with the ranking of the solutions which are involved in it. Thus, we can rewrite Equation (4) as

$$p^s(x_i) = \sum_{\mathbf{x}\setminus x_i} p^s(\mathbf{x}) = \sum_{\tau \in Q_i^{x_i}} p^s(\sigma[\tau]), \tag{6}$$

where $Q_i^{x_i} = \{\tau : \sigma[\tau] = (y_1, \ldots, y_n) \wedge y_i = x_i\}$ is the set of ranking positions corresponding to points $\mathbf{x} \in S$, whose probabilities have been used to calculate the marginal distribution. Note that the cardinality of any $Q_i^{x_i}$ is always 2^{n-1}. For each marginal probability, we have the set of ranking positions associated to $p(X_i = 0)$, denoted by Q_i^0, and the set associated to $p(X_i = 1)$, denoted by Q_i^1. Note that the sets $Q_i^{x_i}$ are intrinsically related to the parameters $\theta_i^{x_i}$. Then, we associate the set $O_i = \{Q_i^0, Q_i^1\}$ with the marginal distribution $p(x_i)$. Note that for all i, $Q_i^0 \cup Q_i^1 = \{1, \ldots, 2^n\}$ and $Q_i^0 \cap Q_i^1 = \emptyset$. In addition, all the sets $Q_i^{x_i}$ involved in the factorization have to be different. Finally, we define the set $G_\sigma = \{O_1, O_2, \ldots, O_n\}$ which includes all the information needed to link the function and the factorization. The ranking positions belonging to each subset depend on the function σ to which the algorithm is applied. In summary, the relationship between the probabilistic model and the function σ is expressed by the structure of sets represented in Figure 2(a). An illustrative example of the definitions mentioned above is presented in Figure 2(b). We can see that by using Equation (6), the

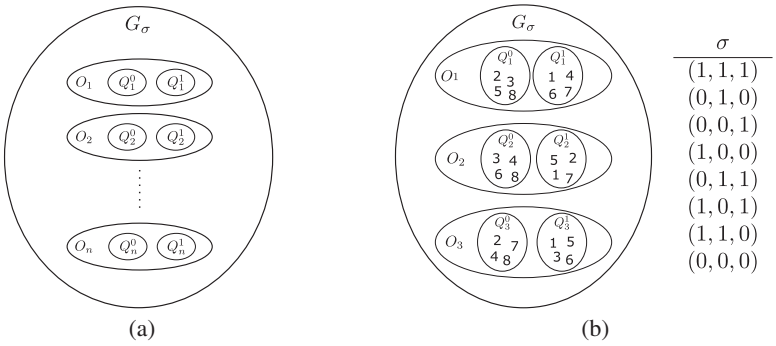


Figure 2: (a) Set-based representation of the relation between the function and the probabilistic model. (b) Example of set G_σ and the corresponding function σ for $n = 3$ variables.

marginal probability $p(X_1 = 0)$ is calculated through the solutions with the rankings $\{2, 3, 5, 8\}$. In turn, $p(X_1 = 1)$ is associated with the set of rankings $\{1, 4, 6, 7\}$. Thus, we have the set $O_1 = \{\{2, 3, 5, 8\}, \{1, 4, 6, 7\}\}$ related to the marginal distribution $p(x_1)$. In this example, we also have the sets $O_2 = \{\{3, 4, 6, 8\}, \{1, 2, 5, 7\}\}$ associated with $p(x_2)$ and $O_3 = \{\{2, 4, 7, 8\}, \{1, 3, 5, 6\}\}$ for $p(x_3)$. Finally, we can create the corresponding set $G_\sigma = \{O_1, O_2, O_3\}$ associated with the factorization of the probability distribution and the function σ being optimized.

In general, we can prove the following necessary and sufficient condition of equivalence between objective functions when algorithm $\mathcal{A}$ implements the function $\mathcal{M}$ according to Equation 3. By using this condition, we will carry out the partition of the space of functions into equivalence classes.

THEOREM 2: *Let $\mathcal{A}$ be an EDA that implements $\mathcal{M}$ as $p^a(\boldsymbol{x}) = \prod_{i=1}^{n} p^s(x_i)$. Two functions σ_1 and σ_2 are equivalent under $\mathcal{A}$ if and only if the corresponding sets G_{σ_1} and G_{σ_2} are equal.*

PROOF: Two functions σ_1 and σ_2 are equivalent if $\mathcal{A}$ generates the same sequence of probability vectors departing from any initial $\mathbf{p}_0$. Therefore, we need to prove that $\mathcal{G}^t(\mathbf{p}_0, \sigma_1) = \mathcal{G}^t(\mathbf{p}_0, \sigma_2)$ in every generation t for all initial $\mathbf{p}_0$ if and only if $G_{\sigma_1} = G_{\sigma_2}$. Firstly, we show that if $G_{\sigma_1} = G_{\sigma_2}$, then $\mathcal{G}^t(\mathbf{p}_0, \sigma_1) = \mathcal{G}^t(\mathbf{p}_0, \sigma_2)$. In order to do that and taking into account that ϕ is independent of σ, it is enough to prove that $\mathcal{M}(\mathbf{p}^s, \sigma_1) = \mathcal{M}(\mathbf{p}^s, \sigma_2)$ for any given $\mathbf{p}^s$. $G_{\sigma_1} = G_{\sigma_2}$ if and only if for any $O_i^{\sigma_1} \in G_{\sigma_1}$, there exists $O_j^{\sigma_2} \in G_{\sigma_2}$ such that $O_i^{\sigma_1} = O_j^{\sigma_2}$ and vice versa. In turn, $O_i^{\sigma_1} = O_j^{\sigma_2}$ if an only if for any $Q_i^{x_i} \in O_i^{\sigma_1}$ there exists $Q_j^{x_j} \in O_j^{\sigma_2}$ such that $Q_i^{x_i} = Q_j^{x_j}$ and vice versa. Therefore, according to Equation (6), if $G_{\sigma_1} = G_{\sigma_2}$, then we are calculating the same set of probability values for both functions and we will obtain the same probability vector $\mathbf{p}^a$ in both cases.

Secondly, we prove that if $\mathcal{G}^t(\mathbf{p}_0, \sigma_1) = \mathcal{G}^t(\mathbf{p}_0, \sigma_2)$ in every generation t for all initial $\mathbf{p}_0$, then $G_{\sigma_1} = G_{\sigma_2}$. To prove this part of the theorem, it suffices to consider a specific set of initial probability vectors containing values greater than 0 only in the desired positions. Thus, let $\mathbf{p}_0 = (p_1, p_2, \ldots, p_{2^n})$ be any initial vector such that $p_i > 0$ if $\sigma_1[i] = (1, x_2, \ldots, x_n)$, otherwise $p_i = 0$. These initial vectors have probability values greater than 0 only in the positions associated with solutions that begin with 1 in σ_1. Due to Property 2 of ϕ, we obtain a vector $\mathbf{p}_0^s$ after selection with nonzero probabilities in the same positions as in $\mathbf{p}_0$. Then, we have that $p^s(X_1 = 1, t = 0) = 1$ for σ_1 after selection.

Table 4: Equivalent functions σ, σ', and σ''.

Rank	σ (x_1, x_2, x_3)	σ' $(\neg x_1, x_2, x_3)$	σ'' $(\neg x_1, x_3, x_2)$
1	(1,1,1)	(0,1,1)	(0,1,1)
2	(0,1,0)	(1,1,0)	(1,0,1)
3	(0,0,1)	(1,0,1)	(1,1,0)
4	(1,0,0)	(0,0,0)	(0,0,0)
5	(0,1,1)	(1,1,1)	(1,1,1)
6	(1,0,1)	(0,0,1)	(0,1,0)
7	(1,1,0)	(0,1,0)	(0,0,1)
8	(0,0,0)	(1,0,0)	(1,0,0)

Since we know that $\mathcal{G}^1(\mathbf{p}_0, \sigma_1) = \mathcal{G}^1(\mathbf{p}_0, \sigma_2)$, it necessarily implies that there exists $j \in \{1, \ldots, n\}$ such that $p^s(X_j = x_j, t = 0) = 1$ for σ_2 and therefore $O_1^{\sigma_1} = O_j^{\sigma_2}$. Otherwise, we would have that $p_i > 0$ for all i in $\mathcal{G}^1(\mathbf{p}_0, \sigma_2)$ and hence, the sequence would be different. By the same argument, any initial vector $\mathbf{p}_0 = (p_1, p_2, \ldots, p_{2^n})$ satisfying $p_i > 0$ if $\sigma_1[i] = (x_1, 1, \ldots, x_n)$ implies that there exists $k \neq j$, $k \in \{1, \ldots n\}$ such that $O_2^{\sigma_1} = O_k^{\sigma_2}$. The same process is repeated for the remaining indices until n, where we consider any initial point $\mathbf{p}_0 = (p_1, p_2, \ldots, p_{2^n})$ such that $p_i > 0$ if $\sigma_1[i] = (x_1, x_2, \ldots, 1)$. Since we have already matched $n - 1$ sets $O^{\sigma_1} \in G_{\sigma_1}$ with the corresponding $n - 1$ sets $O^{\sigma_2} \in G_{\sigma_2}$, there remains a last index variable l such that $O_n^{\sigma_1} = O_l^{\sigma_2}$. Therefore, if the algorithm generates the same sequences of probability vectors from every initial point for σ_1 and σ_2, then $G_{\sigma_1} = G_{\sigma_2}$.

5.2 Characterization of Equivalence Classes

Before addressing in detail the description of the functions belonging to a class, we show in Table 4 an example of three equivalent functions σ, σ', and σ''. We applied two different operations to obtain these equivalent functions. Firstly, we generated the function σ' by negating the values x_1 for all $\mathbf{x} = (x_1, x_2, x_3)$ belonging to σ. Secondly, the function σ'' was obtained by swapping the values x_2 and x_3 for all $\mathbf{x} = (x_1, x_2, x_3)$ belonging to σ'. This operation can be seen as a swapping of the columns x_2 and x_3. In Figure 3, we present the corresponding sets G_σ, $G_{\sigma'}$, and $G_{\sigma''}$ related to the functions of Table 4. Since these sets are equal, we know by Theorem 2 that the functions are equivalent.

In general, taking into account that σ is defined as a 2^n-tuple of the solutions $\mathbf{x} = (x_1, \ldots, x_n) \in S$, the following two operations permit the description of the functions in the class $[\sigma]$:

- **Operator M1 (Negation).** Given a permutation σ, where $\sigma[\tau] = (x_1, \ldots, x_n)$, and an index $i \in \{1, \ldots, n\}$, the operator M1 returns a function σ', where $\sigma'[\tau] = (y_1, \ldots, y_n)$, verifying that $y_i = \neg x_i$ for all ranking positions $\tau \in \{1, \ldots, 2^n\}$. Through this operation we change zeros with ones, and vice versa, in the corresponding variable X_i. This operator can be successively applied until all combinations of variable negations are obtained. Thus, including σ in the count, we can generate 2^n different functions by means of M1.

- **Operator M2 (Swapping).** Given a permutation σ, where $\sigma[\tau] = (x_1, \ldots, x_n)$, and two indexes $i, j \in \{1, \ldots, n\}$, the operator M2 returns a function σ', where

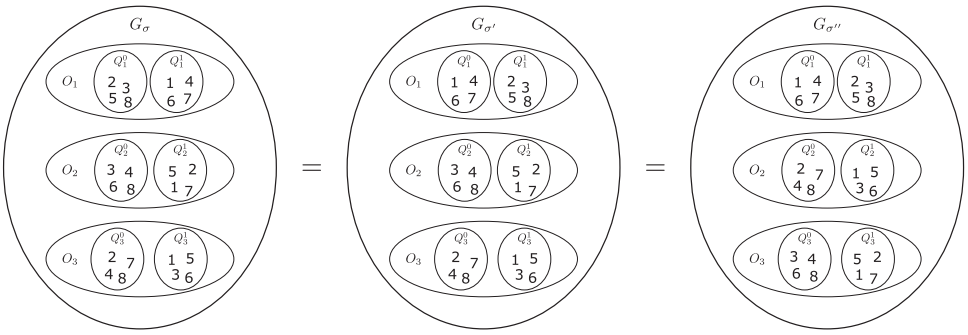


Figure 3: Example with $n = 3$ variables of three equal sets G_σ, $G_{\sigma'}$, and $G_{\sigma''}$ associated with three equivalent functions σ, σ', and σ'', as presented in Table 4.

> $\sigma'[\tau] = (y_1, \ldots, y_n)$, verifying that $y_i = x_j$ and $y_j = x_i$ for all ranking positions $\tau \in \{1, \ldots, 2^n\}$. If we consider the elements of σ disposed in columns, then we can say that the permutation of the columns associated with the variables X_i and X_j produces an equivalent function. From this interpretation, it is easy to see that $n!$ different functions can be generated by applying M2.

According to Theorem 2, we can assert that, given a function σ, the aforementioned operators return equivalent functions σ' because $G_\sigma = G_{\sigma'}$. In fact, these two operators are a consequence of this equality of sets. We have previously shown how the set G_σ is created from σ according to the probabilistic model used by the algorithm. Inversely, we can also read the permutation σ from the set G_σ. Thus, a set $Q_i^{x_i}$ is telling us the positions τ of the permutation σ in which $X_i = x_i$ (cf Figure 2). If these positions are read from every set $Q_i^{x_i}$, then we will obtain σ. Note that if we modify the ranking positions belonging to $Q_i^{x_i}$, we are modifying the positions in which $X_i = x_i$, and therefore we will read a different permutation σ.

Specifically, departing from a given G_σ, we can move the ranking positions that this set contains in two different ways in order to read different functions σ' such that $G_\sigma = G_{\sigma'}$. The first type of movement is as follows. Given any $O_i = \{Q_i^0, Q_i^1\} \in G_\sigma$, the ranking positions $\tau_0 \in Q_i^0$ and $\tau_1 \in Q_i^1$ are swapped together between both subsets to create an equal set $G_{\sigma'}$ in which $\tau_1 \in Q_i^0$ and $\tau_0 \in Q_i^1$. From $G_{\sigma'}$, we can read a new function σ' equivalent to σ. In Figure 3, this type of movement is applied to the set $O_1 \in G_\sigma$ obtaining an equal set $G_{\sigma'}$. From this movement, we deduce that the negation operator M1 produces equivalent functions.

In the second type of movement, given any pair $O_i = \{Q_i^0, Q_i^1\}$, $O_j = \{Q_j^0, Q_j^1\} \in G_\sigma$, we can exchange the ranking positions belonging to both sets O_i and O_j as follows. Let $\tau_i \in Q_i^{x_i}$ and $\tau_j \in Q_j^{x_j}$, we swap the elements between the sets $Q_i^{x_i}$ and $Q_j^{x_j}$ to create the set $G_{\sigma'}$ in which $\tau_j \in Q_i^{x_i}$ and $\tau_i \in Q_j^{x_j}$. All the ranking positions τ_i belonging to $O_i = \{Q_i^0, Q_i^1\}$ have to be moved at the same time, otherwise, the associated marginal probability will make no sense. In Figure 3, this type of movement is applied to the sets O_2 and O_3 in $G_{\sigma'}$ producing an equal set $G_{\sigma''}$. From this movement we deduce that the swapping operator M2 produces equivalent functions.

Based on the abovementioned movements that the equality of sets allows, it can be stated that, given σ, the operators M1 and M2 allow a description of all the functions in the class $[\sigma]$. There are no more operations that produce equivalent functions according

to Theorem 2. Therefore, by combining all possible functions that can be generated by using M1 and M2, we can conclude that the number of equivalent functions σ per class is $n!2^n$. Therefore, the number of classes is

$$\frac{2^n!}{n!2^n} = \frac{(2^n-1)!}{n!}.$$

This is the number of different behaviors that a univariate EDA can show in solving Equation (1).

The operators of negation and swapping presented in this section are closely related to the xor-invariance and the permutation-invariance introduced in Lehre and Witt (2012) in order to conduct time complexity analysis of randomized search heuristics using unbiased variation operators. Although the operators presented in the current paper and the properties defined in Lehre and Witt have technical differences, both approaches provide a scenario in which the algorithm is not biased by the specific configuration of the solutions. In the terms used in Lehre and Witt, and setting aside the differences, we could say that the univariate EDA is an unbiased algorithm.

5.3 Equivalence Classes and Local Optima

The impact that different problem characteristics have in the performance of search algorithms, and hence in the difficulty of the problem, is a fundamental topic in the optimization field. Properties such as number of local optima, additive decomposability of the function (Mühlenbein and Mahnig, 1999; Gao and Culberson, 2005), fitness landscape network (Liu et al., 2012), and different difficulty measures (Naudts and Kallel, 2000; Liu et al., 2012) have been proposed and studied in order to advance the performance of EAs. In fact, the problem difficulty analysis is also useful to classify optimization problems (Jansen, 2001). Although this type of classification is performed in a very different way from the taxonomy of problems presented in the current paper, it could be possible to create links between both approaches. Thus, the equivalence classes can be connected to characteristics of the problems belonging to them with the aim of assisting in the prediction of problem difficulty for EDAs.

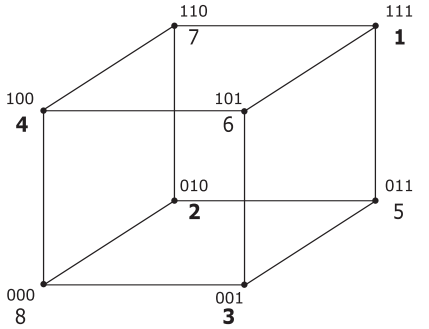
Particularly, in the field of EDAs, the relation between the local optima of the function and specific EDA implementations has been theoretically shown in González et al. (2001) and Zhang (2004). In this section, we connect this property of the problems with the equivalence classes developed along the paper. We consider the neighborhood system in S induced by the Hamming distance. Thus, the distance $H(\mathbf{x}, \mathbf{y})$ between two solutions $\mathbf{x}$ and $\mathbf{y}$ is given by

$$H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|.$$

The neighbors of a solution $\mathbf{x}$ are those solutions $\mathbf{y} \in S$ such that $H(\mathbf{x}, \mathbf{y}) = 1$. In terms of the permutation σ, a solution $\mathbf{x}$ is called a local optima if $\sigma(\mathbf{x}) < \sigma(\mathbf{y})$ for any $\mathbf{y} \in S$ such that $H(\mathbf{x}, \mathbf{y}) = 1$. Figure 4 illustrates how the neighborhood system and the local optima can be seen for the given σ. Each vertex of the cube represents a solution $\mathbf{x} \in S$. Below each solution, we have the corresponding position of each $\mathbf{x}$ in σ. It can be checked that the first four solutions in the permutation are local optima because they are better solutions than their neighbors. These solutions are marked in bold.

The relation between the equivalence classes and the Hamming neighborhood system in S is established by the following theorem.

THEOREM 3: *All the functions in the same equivalence class $[\sigma]$ have the same number of local optima and in the same ranking positions.*



σ
$(1,1,1)$
$(0,1,0)$
$(0,0,1)$
$(1,0,0)$
$(0,1,1)$
$(1,0,1)$
$(1,1,0)$
$(0,0,0)$

Figure 4: Example of Hamming neighborhood system with $n = 3$ over the function σ. Numbers in bold represent local optima.

PROOF: A local optimum can be defined according to the position that it holds in the permutation σ and the Hamming distance to the preceding solutions. Then, a solution $\sigma[i] = \mathbf{x}$ is a local optima for σ if $H(\sigma[i], \sigma[j]) \geq 2$ for all $j < i$. Given a function σ, it is easy to see that by applying the operators M1 and M2 of negation and swapping, any function σ' that we obtain verifies $H(\sigma[i], \sigma[j]) = H(\sigma'[i], \sigma'[j])$ for all $i, j \in \{1, \ldots, 2^n\}$. Therefore, if there is a local optima in the position i of σ, then there is a local optima in the same position i of σ', and vice versa. Since the operators M1 and M2 allow for a description of all the functions of a class, these functions always have the same number of local optima and are in the same ranking positions. □

From the results presented in Lehre and Witt (2012), it is also possible to deduce that arbitrary applications of M1 and M2 preserve Hamming distance. This is due to the property of Hamming-invariance that the unbiased variation operators satisfy.

Theorem 3 agrees with the results presented by González et al. (2001) and Zhang (2004). According to these papers, all the local optima are attractive fixed points (Scheinerman, 1996) of the dynamical systems that were used to model the corresponding univariate EDA implementations. Therefore, since we say that the algorithm has the same behavior for all the functions belonging to a class, all those functions should necessarily have the same number of local optima and in the same ranking positions to support González et al. and Zhang. Nevertheless, Theorem 3 also provides a more general perspective because it implies that the relation between univariate EDAs and the local optima of the function is an intrinsic property of the probabilistic model and is independent of the implementation of the selection scheme.

6 Numerical Experiments in $S = \{0, 1\}^3$

In this section, we use the previously elaborated partition of classes of the functions to carry out a more detailed analysis of the different behaviors the univariate EDA can have in the injective functions in $\{0, 1\}^3$. Note that conducting numerical experiments to analyze the behavior of the algorithm in all equivalence classes is only feasible for very small n (e.g., $n \in \{3, 4\}$). For greater n, conducting an exhaustive analysis of the different EDA behaviors becomes intractable and then only theoretical studies or specific experiments can be carried out. We conduct numerical simulations of an EDA with infinite population which implements tournament selection and computes the approximation step according to Equation (5). The local dynamical behavior of this algorithm was theoretically studied in Zhang (2004).

Particularly, we will concentrate on the complexity of the classes for the algorithm. The complexity is measured by two descriptors: (i) the size of the basin of attraction of the global optimum, and (ii) the generated sequence of probability vectors. We will consider that the smaller the size of the basin of attraction in a class, the more difficult the problems in that class are. Moreover, for two classes with the same basin size, the more time the algorithm takes to converge, the more difficult the function is.

In order to add more information to this analysis, and taking into account the relevant role that the local optima play in the EDA that assumes independence (see Theorem 3), we will put the previous complexity results in relation to this problem characteristic. We will see that the complexity for the EDA in terms of the size of the basin of attraction is closely related to the number of local optima and their positions in the function ranking. To the best of our knowledge, this is the first time for such an analysis to appear in the literature.

6.1 Experimental Design

We take into account all the possible σ that can be constructed over the search space $S = \{0, 1\}^3$. Therefore, we consider $2^3! = 40320$ functions. By creating the set G_σ for each function and applying Theorem 2 for each pair of sets, we group the functions by equivalence classes. We have $3!2^3 = 48$ functions per class and hence 840 classes. We only need to consider one function per class because all the functions in the same class behave equivalently. The selection of the function which represents the class is arbitrary.

To carry out the EDA simulations, we need to specify four elements: the initial points, the selection mechanism, the approximation step (Equation (5)), and the stopping condition. We create 10,000 initial probability vectors which try to be representative of the simplex Ω_8. These initial points have been randomly generated by sampling a Dirichlet distribution with all the parameters equal to 1. In addition, we also take into account the uniform distribution as an initial point. Then, for each function, we launch 10,001 EDA runs, one from each initial probability vector previously generated. All these runs try to represent the EDA behavior for the corresponding optimization problem.

We use two-tournament selection according to Zhang (2004) to implement the selection ϕ. This selection takes uniformly at random two solutions of the population and then chooses the individual with the best objective function value. This procedure should be repeated until the selected set is completed (Algorithm 1). Since we deal with injective functions, two solutions cannot have the same function value. In the infinite population EDA model, the probability vector $\mathbf{p}^s$ after tournament selection can be computed from the vector $\mathbf{p}$ as follows:

$$p_i^s = p_i^2 + 2p_i \sum_{j=i+1}^{n} p_j. \tag{7}$$

Tournament selection obeys the properties that we imposed to ϕ in Section 3.2. Therefore, the partition induced from the equivalence condition of Theorem 2 is valid.

The stopping condition of the algorithm is a maximum of 40 iterations. This number of generations provides a satisfactory trade-off between accuracy in the numerical results and computational cost. The algorithm converges to 1 in 93% of all the runs conducted. In the rest of the runs, the highest probability value after 40 generations is always greater than 0.9998. In these cases, it is assumed that the algorithm has converged to the corresponding solution. The numerical precision that we have used is double-precision floating point, requiring 64 bits per stored value.

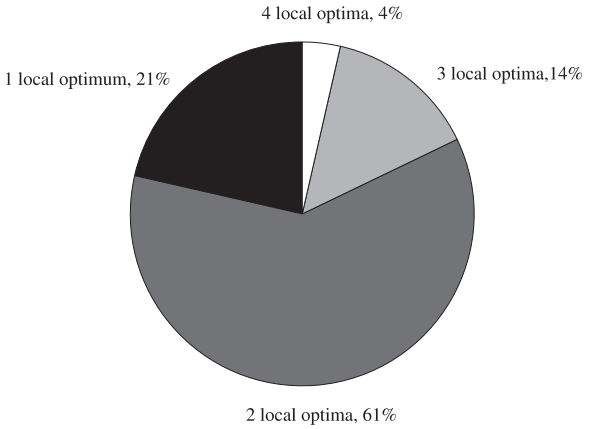


Figure 5: Proportion of classes with different numbers of local optima.

In the numerical analysis, the size of the basins of attraction can be stored in a vector $\mathbf{b} = (b_1, \ldots, b_8)$ where each b_i is the number of initial points that have converged to the solution with rank i.

6.2 Results

First of all, in Figure 5, we show the proportion of functions with different numbers of local optima in order to provide a general perspective of this problem characteristic. By considering classes instead of specific functions, we have 180 classes with just one local optimum (the global optimum), 510 classes with two local optima, 120 classes with three local optima, and finally 30 classes with four local optima.

6.2.1 Analysis Based on Basins of Attraction

In order to provide a first general picture of the equivalence classes, we represent in Figure 6 the basins of attraction of the optimum by means of different colors. The sizes of these basins of attraction are interpreted in terms of problem difficulty. The color bar on the right of this picture indicates the relation between the colors and the size of the basin. At the top of the spectrum, the white color is assigned to the largest basins of attraction which indicate easy problems. At the bottom, the dark colors represent low basins and hence, they reveal the hardest problems. In addition, the classes have been grouped by the number of local optima. Thus, the picture has been divided into four parts separated by vertical dashed lines. From left to right, we have the classes with one, two, three, and four local optima, respectively. In each of these parts, the classes are ordered according to the size of the basins of attraction.

Note that we have assigned the white color only to the classes in which the size of the basins is 10,001 or is a very close number to that. We start to use light gray colors when approximately 150 initial points do not converge to the optimum. We try to highlight all the small variations between classes because they could imply dramatic differences in EDAs with finite populations and problems with greater dimension. According to Figure 6, we could say that the white classes are easy. These white classes cover all the problems with one local optima and a small number of problems with two local optima. It can be observed that a higher number of local optima does not necessarily imply more difficult problems. In fact, the darkest colors are in the area corresponding

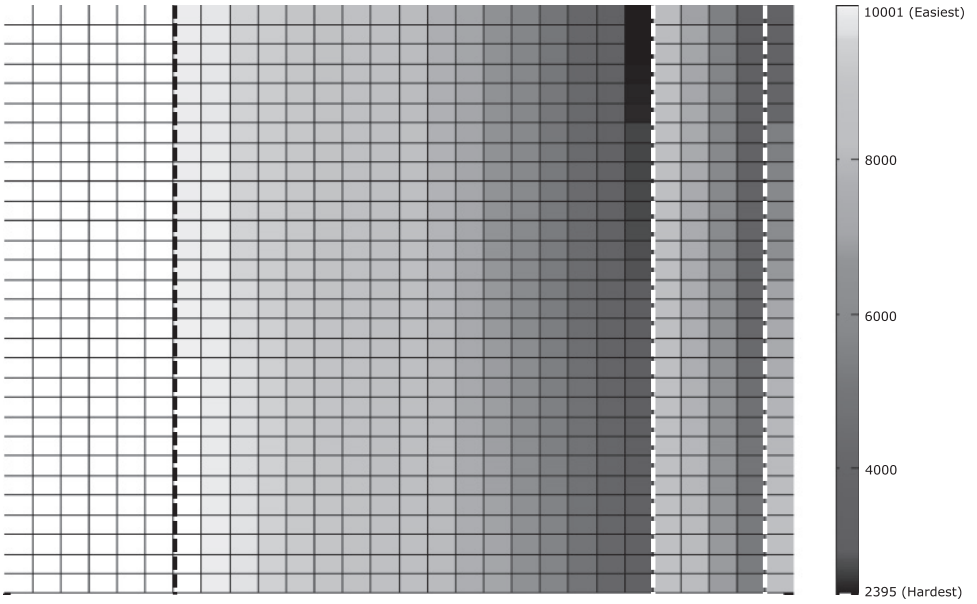


Figure 6: Size of the basin of attraction of the optimal solution for each class.

to classes with two local optima. It is the zone in which we can see the widest range of colors.

As discussed in Section 4, different classes can have the same basins of attraction. Thus, Figure 6 shows how the classes can be grouped according to these basins of attraction. This fact could be related to the existence of a second level of grouping among classes. Nevertheless, it deserves a specific study.

Through Figure 7, we analyze the basins of attraction of both the global optimum and the rest of local optima. We consider all the ranking positions at which the local optima can be allocated. In this case, when we refer to local optima, the global optimum is not included. The different scenarios are indicated in the legends and expressed by means of different markers. Note that in these plots only the basins of the global optimum and the best local optimum are explicitly indicated. The dashed line $y = 10001 - x$ is used as a reference to illustrate the proportion of initial points that have converged to the rest of the local optima.

In Figure 7(a), we show the classes with one local optimum. This solution can be in different positions of the permutation σ as indicated in the figure. Note that the difficulty of the problem strongly depends on the ranking position of the local optimum. When this solution changes from the second position to the third position of the ranking, the difficulty of the problem decreases dramatically. In fact, when the local optimum has rank 5, the complexity of the problems is very similar to the complexity of a problem without local optima. Figures 7(b) and 7(c) show the classes with two and three local optima, respectively. We can see how the final convergence of the algorithm is distributed among the different local optima. Analogously to the previous situation, as the local optima have a lower rank, their basins of attraction clearly decrease. Particularly, when the rank of the worst local optimum changes from 3 to 4 in Figure 7(b), the difficulty of the problems dramatically decreases. In Figure 7(c) the changes are more subtle. Nevertheless, we can observe that the classes with a local optimum in the fifth

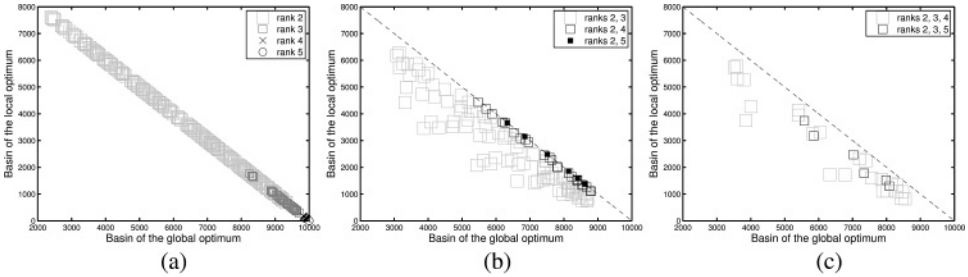


Figure 7: Basins of attraction of the optimal solution (x axis) and the best local optimum (y axis). (a) Classes with one local optimum besides the global optimum. As indicated in the legend, the local optimum can hold the ranks 2, 3, 4, or 5 in a permutation σ. (b) Classes with two local optima besides the global optimum. The two local optima can be at the following ranking positions: $(2, 3)$, $(2, 4)$, and $(2, 5)$. (c) Classes with three local optima besides the global optimum. The three local optima can be at the following ranking positions: $(2, 3, 4)$ and $(2, 3, 5)$. The dashed line ($y = 10001 - x$) serves as a reference to indicate the basins of attraction of the rest of the local optima.

position never reach the highest complexities of the classes with a local optimum in the fourth position.

6.2.2 Analysis Based on Sequences

We know that the sequences of probability vectors generated by the algorithm uniquely identify the functions in a class. In this section, we use this fact in order to distinguish the different EDA behaviors for the classes with only one local optimum (the global optimum). According to the basins of attraction (Figure 6), all these classes have the same complexity for the algorithm. However, in Figure 8 we can observe different convergence behaviors. In Figure 8(a), we show the curves that the probability of the optimum depicts throughout the generations. Alternatively, Figure 8(b) represents the same probabilities of the optimum by means of colors. The specific probability values are shown in the color bar on the right. Particularly in Figure 8(b), we can clearly see how several classes on the bottom of the chart have a slower convergence. This slower convergence to the optimum can also be interpreted as a consequence of facing harder problems. From this point of view, we could say that not all the functions with one local optimum have the same complexity. This fact could have implications in EDAs with finite samples.

7 Discussion and Future Work

There are a number of directions in which we plan to extend the results presented in this paper. In the following paragraphs, the most important points are discussed.

An important generalization is the introduction of noninjective functions in the analysis. In general terms, instead of dealing with only one σ to represent a function, we could consider a set $\{\sigma\}$ of permutations to represent noninjective functions. An adequate definition and management of noninjectivity in algorithm $\mathcal{A}$ would be essential in order to extend Theorems 1 and 2 to this type of function.

The function $\mathcal{M}$ in algorithm $\mathcal{A}$ can be extended to deal with a set of different factorizations; or this operator could be restricted to work with a maximum complexity for the probabilistic model. By means of this generalization, the algorithm could

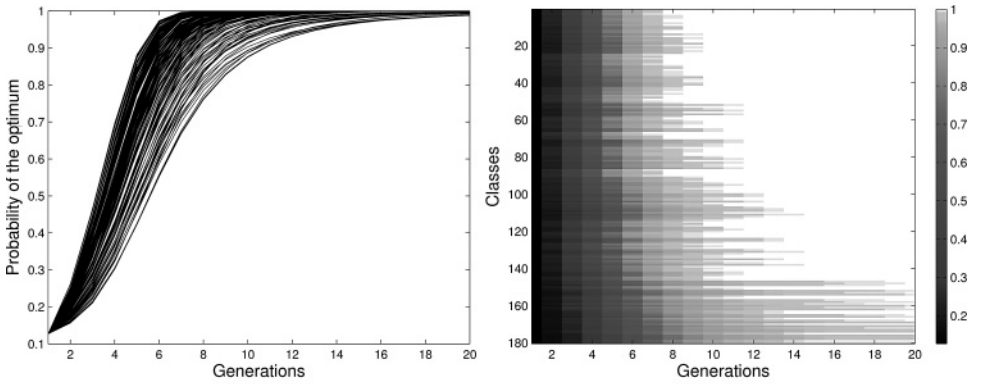


Figure 8: Probability of the optimum along the generations for the classes with only one local optimum. (a) Curves of the probability of the optimum. (b) In the electronic version, the probability of the optimum represented by means of colors for each class. The specific probability values are shown in the color bar.

represent the behaviors of complex EDAs such as those that implement learning of Bayesian networks.

The sets G_σ can also be extended to include any type of factorization expressed by means of marginal and conditional probability functions. In this regard, it would be necessary to develop new properties to guarantee valid equivalences and to characterize equivalent functions. An interesting related issue is to know how the number of classes decreases as the complexity allowed in the probabilistic model increases.

In turn, the definition of equivalence can be directly extended to include noninjective functions or more complex algorithms because, in essence, we only demand equal sequences of probability vectors. This is independent of the way in which the noninjectivity or the probabilistic models are managed by the algorithm.

In relation to the selection ϕ, we distinguish the following two important open issues. Firstly, alternative definitions of equivalence between problems should be developed to include selection schemes that take into account the specific function values of the solutions. Secondly, a crucial point is to take into account the convergence of the algorithm to the optimal solution. In order to deal with convergence issues in the framework of the equivalence classes, we need to take into account the properties that the selection ϕ should obey (Section 3).

In the current paper, we have shown the connection between the equivalence classes and the neighborhood system induced by Hamming distance for univariate EDAs. We hypothesize that it is possible to discover other links with problem characteristics and descriptors. The equivalence classes can be connected, on the one hand, with problem descriptors and, on the other hand, with the difficulty of the problems. Therefore, the information that we could have about the problem at hand can be used to identify the class to which it belongs and then try to advance how the algorithm will perform.

8 Conclusions

This work can be divided into three main parts. Firstly, we have laid the foundations to create taxonomies of problems under EDAs by providing the needed definitions regarding the optimization problems, the algorithm, and the equivalence relation. From

these definitions, it has been deduced that all the problems are in the same class when the probabilistic model does not impose restrictions to approximate the distribution of the selected individuals. Secondly, we have studied the taxonomy of problems that arises under a univariate EDA. To express the relation between the probabilistic model and the function, we have defined the sets G_σ. Based on these sets, we are able to provide a necessary and sufficient condition to decide the equivalence between functions and to partition the space of problems. Through the operators of negation and swapping, it is possible to describe all the functions in a class and count its members. By taking into account the aforementioned elements, we reveal an intrinsic connection between the univariate probabilistic model and the neighborhood system induced by Hamming distance. In the third and last part, we have conducted numerical simulations of a univariate EDA which implements tournament selection. We can extract the following main conclusions from the experiments. (i) A higher number of local optima does not necessarily imply more difficult problems. In this regard, the difficulty of the problem strongly depends on the ranking position of the local optima. (ii) We have observed how the classes can be grouped according to the basins of attraction showing a second level of grouping. (iii) We have shown that not all the functions without local optima have the same complexity.

In general terms, this paper introduces a framework to formally study the relationship between EDAs and the space of optimization problems. The results that we have presented can be generalized and extended in many directions. Specifically, once the partition of the space of problems has been created, we consider the following questions particularly important:

1. How can we describe and identify the classes of easy and hard problems for EDAs? (Chen et al., 2010)
2. Which are the problem descriptors that allow to identify the class to which that problem belongs to?
3. How can we study the convergence of the algorithm (Zhang, 2004; Zhang and Mühlenbein, 2004) for the problems in a class?
4. Which is the minimum complexity that should be introduced in the probabilistic model in order to converge to the optimum? (Echegoyen et al., 2012)
5. How can we extrapolate these results to algorithms that use finite samples?

These types of issues could be translated to any EA. We believe that working in the direction given by these questions is important to reach an in depth understanding of the underlying mechanisms that govern EAs.

Acknowledgments

This work was partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2010-14931 (Spanish Ministry of Science and Innovation), and COMBIOMED network in computational biomedicine (Carlos III Health Institute). Carlos Echegoyen holds a fellowship from UPV-EHU.

References

Armañanzas, R., Inza, I., Santana, R., Saeys, Y., Flores, J. L., Lozano, J. A., Van de Peer, Y., Blanco, R., Robles, V., Bielza, C., and Larrañaga, P. (2008). A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, 1(6):1–12.

Blickle, T., and Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394.

Bosman, P. A. (2010). The anticipated mean shift and cluster registration in mixture-based EDAs for multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2010*, pp. 351–358.

Brownlee, A., Pelikan, M., McCall, J., and Petrovski, A. (2008). An application of a multivariate estimation of distribution algorithm to cancer chemotherapy. In *Proceedings of the 10th Genetic and Evolutionary Computation Conference, GECCO-2008*, pp. 1033–1040.

Chen, T., Tang, K., Chen, G., and Yao, X. (2010). Analysis of computational time of simple estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):1–22.

Doerr, B., and Winzen, C. (2012). Ranking-based black-box complexity. *Algorithmica*. To appear.

Echegoyen, C., Mendiburu, A., Santana, R., and Lozano, J. A. (2012). Towards understanding EDAs based on Bayesian networks through a quantitative analysis. *IEEE Transactions on Evolutionary Computation*, 16(2):173–189.

Echegoyen, C., Zhang, Q., Mendiburu, A., Santana, R., and Lozano, J. A. (2011). On the limits of effectiveness in estimation of distribution algorithms. In *Proceedings of the 2011 Congress on Evolutionary Computation, CEC-2011*, pp. 1573–1580.

Gao, Y., and Culberson, J. C. (2005). Space complexity of estimation of distribution algorithms. *Evolutionary Computation*, 13(1):125–143.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

González, C., Lozano, J. A., and Larrañaga, P. (2001). Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, 12(4):465–479.

González, C., Lozano, J. A., and Larrañaga, P. (2002). Mathematical modeling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions. *International Journal of Approximate Reasoning*, 31(4):313–340.

Hauschild, M., Pelikan, M., Sastry, K., and Goldberg, D. E. (2012). Using previous models to bias structural learning in the hierarchical BOA. *Evolutionary Computation*, 20(1):135–160.

Hauschild, M., Pelikan, M., Sastry, K., and Lima, C. (2009). Analyzing probabilistic models in hierarchical BOA. *IEEE Transactions on Evolutionary Computation*, 13(6):1199–1217.

Jansen, T. (2001). On classifications of fitness functions. *Theoretical aspects of evolutionary computing* (pp. 371–385). Berlin: Springer-Verlag.

Larrañaga, P., and Lozano, J. A. (Eds.). (2002). *Estimation of distribution algorithms. A new tool for evolutionary computation*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Lee, K. Y., and El-Sharkawi, M. A. (Eds.). (2008). *Modern heuristic optimization techniques: Theory and applications to power systems*. New York: John Wiley & Sons.

Lehre, P. K., and Witt, C. (2012). Black-box search by unbiased variation. *Algorithmica*. To appear.

Liu, J., Abbass, H. A., Green, D. G., and Zhong, W. (2012). Motif difficulty (MD): A predictive measure of problem difficulty for evolutionary algorithms using network motifs. *Evolutionary Computation*, 20(3):321–347.

Mühlenbein, H., and Mahnig, T. (1999). FDA—A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376.

Mühlenbein, H., Mahnig, T., and Ochoa, A. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):213–247.

Mühlenbein, H., and Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. In *Parallel problem solving from nature, PPSN IV. Lectures notes in computer science*, Vol. 1141 (pp. 178–187). Berlin: Springer-Verlag.

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

Pelikan, M. (2005). *Hierarchical Bayesian optimization algorithm. Toward a new generation of evolutionary algorithms*. Berlin: Springer-Verlag.

Prügel-Bennett, A. (2000). Finite population effects for ranking and tournament selection. *Complex Systems*, 12(2):183–205.

Santana, R., Larrañaga, P., and Lozano, J. A. (2008). Protein folding in simplified models with estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 12(4):418–438.

Santana, R., Larrañaga, P., and Lozano, J. A. (2009). Research topics on discrete estimation of distribution algorithms. *Memetic Computing*, 1(1):35–54.

Scheinerman, E. R. (1996). *Invitation to dynamical systems*. Englewood Cliffs, NJ: Prentice-Hall.

Shapiro, J. L. (2005). Drift and scaling in estimation of distribution algorithms. *Evolutionary Computation*, 13(1):99–123.

Vose, M. D. (1999). *The simple genetic algorithm: Foundations and theory*. Cambridge, MA: MIT Press.

Zhang, Q. (2004). On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 8(1):80–93.

Zhang, Q., and Mühlenbein, H. (2004). On the convergence of a class of estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):127–136.