Theoretical Computer Science 598 (2015) 64–86

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Comprehensive characterization of the behaviors of estimation of distribution algorithms

Carlos Echegoyen *, Roberto Santana, Alexander Mendiburu, Jose A. Lozano

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

ARTICLE INFO

Article history:
Received 16 March 2014
Received in revised form 6 February 2015
Accepted 14 April 2015
Available online 20 April 2015
Communicated by B. Doerr

Keywords:
Heuristic optimization
Machine learning
Estimation of distribution algorithms
Factorizations
Equivalence classes
Neighborhood systems

ABSTRACT

Estimation of distribution algorithms (EDAs) are a successful example of how to use machine learning techniques for designing robust and efficient heuristic search algorithms. Understanding the relationship between EDAs and the space of optimization problems is a fundamental issue for the successful application of this type of algorithms. A step forward in this matter is to create a taxonomy of optimization problems according to the different behaviors that an EDA can exhibit. This paper substantially extends previous work in the proposal of a taxonomy of problems for univariate EDAs, mainly by generalizing those results to EDAs that are able to deal with multivariate dependences among the variables of the problem. Through the definition of an equivalence relation between functions, it is possible to partition the space of problems into equivalence classes in which the algorithm has the same behavior. We provide a sufficient and necessary condition to determine the equivalence between functions. This condition is based on a set of matrices which provides a novel encoding of the relationship between the function and the probabilistic model used by the algorithm. The description of the equivalent functions belonging to a class is studied in depth for EDAs whose probabilistic model is given by a chordal Markov network. Assuming this class of factorization, we unveil the intrinsic connection between the behaviors of EDAs and neighborhood systems defined over the search space. In addition, we carry out numerical simulations that effectively reveal the different behaviors of EDAs for the injective functions defined over the search space $\{0,1\}^3$. Finally, we provide a novel approach to extend the analysis of equivalence classes to non-injective functions.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Recently, there has been an increasing interest in the application of machine learning techniques in combinatorial and stochastic optimization [1–4]. The use of machine learning in optimization provides a more efficient exploitation and representation of the information gathered about the search space. One consolidated example of the methods that incorporate machine learning to optimization are estimation of distribution algorithms (EDAs) [5–7]. Strong evidence of their popularity is the development of new and more complex EDAs [8–10], their application both to real and academic problems [11–13] and the studies of fundamental issues in order to better understand how these algorithms perform [14–16].

* Corresponding author.
E-mail addresses: carlos.echegoyen@ehu.es (C. Echegoyen), roberto.santana@ehu.es (R. Santana), alexander.mendiburu@ehu.es (A. Mendiburu), ja.lozano@ehu.es (J.A. Lozano).

http://dx.doi.org/10.1016/j.tcs.2015.04.015
0304-3975/© 2015 Elsevier B.V. All rights reserved.

EDAs are a class of evolutionary algorithms (EAs) [17]. Commonly, EAs search for the best solutions of a problem by maintaining a population of individuals (solutions) that evolves from one generation to the next. The evolution is carried out by selecting a subset of promising individuals from the population and applying recombination operators that create new individuals (with the aim of obtaining higher quality solutions at each step). In particular, EDAs use machine learning methods to extract relevant features of the search space through the selected individuals of the population. The learning algorithm allows to estimate a new probability distribution over the search space at each step of the EDA. Thus, each of the candidate solutions has an associated probability of being sampled, which varies during the optimization process. 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, in practice, it is hard to know to what extent the EDA is working as desired.

In general, although the application of EDAs can be relatively easy, predicting any aspect of their behavior or determining the quality of the search are non-trivial matters. This scenario motivates the study of the underlying mechanisms that govern EDAs in order to better understand their behavior when solving problems. For instance, there are works that analyze the convergence of specific EDAs to the global optimum and the local optima under ideal conditions [14,18,19], works that analyze the time complexity [20] of univariate EDAs in relation to the problem size or other works that conduct EDA runs on benchmark problems to study the structural models learned by the algorithm [21], or the probability values generated during the search [22]. The current work, which provides new knowledge in this research field, presents an exhaustive classification of the different behaviors that certain EDAs can exhibit, and studies how these behaviors are related with the local optima of the optimization problems.

The behavior of an EDA, when it is applied to a given objective function, can be accurately described by the sequence of probability distributions generated at each generation. We use this description of the EDA in order to identify and classify the different behaviors. Given the complexity of this task, a number of assumptions have to be made: i) we consider EDAs with infinite populations (although the taxonomy of problems developed throughout the paper is also valid for finite populations), ii) the selection scheme is based on the rank of the solutions and iii) the algorithm is applied in the space of injective functions. By considering infinite populations it is possible to see the EDA as a deterministic dynamical system. This type of EDA avoids the random errors caused by sampling finite sets of solutions, which is desirable for certain theoretical analysis. Section 3 explains in more detail this type of EDA.

A crucial element in this work 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 prove that the EDA exhibits the same performance for all the functions belonging to the same class and that the performance of the algorithm is different in each class. Thus, it is possible to group the optimization problems according to the behavior of the algorithm, creating a taxonomy.

This novel research line was initiated in [23], where the study is restricted to EDAs that assume independence among the variables of the problem. In the current work, the study is generalized to EDAs that are able to deal with multivariate dependences among the variables of the problem. In order to present a self-contained paper, we will need to include some definitions and explanations already included in [23]. Nevertheless, the current work provides significant advances which are summarized as follows. Firstly, in order to create partitions of the space of problems, the necessary and sufficient condition to identify equivalent functions that was presented in [23] is generalized here to EDAs that implement any factorization given by a Bayesian network [24]. Secondly, based on this condition, we carry out a novel detailed description of the functions belonging to a class and count the number of equivalent functions. Due to the wide casuistry that the use of general Bayesian networks entails, we have restricted the analysis to factorizations given by chordal Markov networks, which are a subclass of Bayesian networks. We argue that this type of models are widely used in EDAs. For instance, they are present in the earliest EDAs [25], in current research lines of EDAs [26] and also in theoretical works [27]. Thirdly, we study the connection between the equivalence classes generated by an EDA that implements a chordal Markov network and the local optima of the functions belonging to each class. In order to do that, we define a distance associated to the factorization implemented by the EDA. Then, according to this distance, we show that the functions in the same class have the same number of local optima and in the same ranking positions. In addition, we show that the algorithm cannot converge to a solution which is not a local optimum. These facts reveal the intrinsic connection between neighborhood systems and EDAs and open the path for the implementation of informed neighborhood search schemes within the EDA that take advantage of the factorization for the definition and exploitation of neighborhood systems. Finally, we conduct extensive numerical simulations of the EDAs that introduce a Bayesian network of three variables. These EDAs implement tournament selection [14] and they are applied to the injective functions defined over the search space $\{0, 1\}^3$. Through the numerical analysis, we complement the theoretical study by analyzing the complexity of the problems belonging to each class. The difficulty of the problems is presented in relation to the local optima due to the relevant role that this problem descriptor plays in EDAs.

The rest of the paper is organized as follows. Section 2 formally introduces optimization problems and presents EDAs. Section 3 describes the EDAs with infinite populations considered in the current work. In Section 4, the concept of equivalence between functions is presented and discussed. Section 5 introduces the factorizations given by Bayesian networks and the equivalence condition. In Section 6 an in-depth description of the functions belonging to a class is carried out. Section 7 studies the relationship between EDAs and local optima. Section 8 presents numerical experiments to provide complemen-

Table 1
Example of ranking π induced by an injective function for $n = 3$.

$f(\boldsymbol{x})$	π	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

tary information and illustrates the previous theoretical results. In Section 9 a novel approach to the analysis of equivalence classes for non-injective functions is presented. Section 10 draws the conclusions obtained during the study.

2. Background

2.1. Optimization problems

In this paper we consider the following general optimization problem:

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

where $\mathcal{X}$ is a discrete search space, $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}$ is a solution and $f : \mathcal{X} \to \mathbb{R}$ is the objective function. For the sake of simplicity, we assume binary variables although the main results provided do not depend on the cardinality of the variables or the codification of the solutions. The cardinality $|\mathcal{X}|$ of the search space is therefore 2^n. Throughout the paper, we assume that the function $f(\boldsymbol{x})$ is injective. Thus, for each $z \in f(\mathcal{X})$ there is only one $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}$ such that $f(\boldsymbol{x}) = z$.

Since $f(\boldsymbol{x})$ is injective, it naturally induces an explicit ranking π of the solutions of the search space $\mathcal{X}$. π is defined as a mapping from $\{1, \ldots, |\mathcal{X}|\}$ to $\mathcal{X}$. This ranking π is ordered according to the function values $f(\mathcal{X})$. Thus, the solution $\pi(1)$, namely $\boldsymbol{x}_1$, has the highest function value, $\pi(2) = \boldsymbol{x}_2$ has the second highest and so on, with the last solution $\pi(|\mathcal{X}|) = \boldsymbol{x}_{|\mathcal{X}|}$ being the one with the lowest function value:

$$f(\pi(1)) > f(\pi(2)) > \ldots > f(\pi(|\mathcal{X}|)).$$

Thus, we interpret that $\pi(i) = \boldsymbol{x}$ returns the solution $\boldsymbol{x} \in \mathcal{X}$ which is at position i in the ranking. The solution $\pi(1)$ corresponds to the solution $\boldsymbol{x}^*$ that solves Eq. (1). Independently of the specific function values $f(\mathcal{X})$, whenever they provide the same ranking of the solutions $\boldsymbol{x} \in \mathcal{X}$, the function is represented by the same ranking π. In the case of injective functions, we have $|\mathcal{X}|!$ different rankings and the set containing all possible π is denoted by Π. From now on, the objective function $f(\boldsymbol{x})$ and the corresponding π can be used indistinctly as synonyms.

In Table 1, we provide an example of the ranking π induced by a function $f(\boldsymbol{x})$. In the first column, the original function values are shown. In the second column, we represent the solutions according to the ranking π and in the third column, we explicitly indicate the rank i 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) [28]. Instead of exchanging information between individuals through genetic operators, EDAs use machine learning methods [6] to extract relevant features of the search space through the selected individuals of the population. The collected 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 $\boldsymbol{x}^*$. Therefore, we need the following definitions. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a vector of n binary random variables. We define $p(\mathbf{X} = \boldsymbol{x})$, or simply $p(\boldsymbol{x})$, as the joint probability distribution of the variables $\mathbf{X}$ taking values from the search space S. Let $I = \{1, \ldots, n\}$ and $K \subseteq I$. The symbol $\boldsymbol{x}_K$ denotes a vector containing the values $(x_i)_{i \in K}$ and $p(\boldsymbol{x}_K)$ is the corresponding marginal probability function of $p(\boldsymbol{x})$. The general scheme of an EDA is shown in Algorithm 1.

It is convenient to explain the EDA in terms of the probability distributions involved in the search [29] (see Fig. 1). Firstly, D_t denotes the EDA population at generation t and $p(\boldsymbol{x}, t)$ is the underlying probability distribution of this sample. Secondly, $p^s(\boldsymbol{x}, t)$ is the probability distribution of the selected individuals D_t^s. Finally, $p^a(\boldsymbol{x}, t)$ is the distribution given by the probabilistic model chosen to approximate $p^s(\boldsymbol{x}, t)$. Once we have $p^a(\boldsymbol{x}, t)$, the next generation D_{t+1} is constructed by sampling this distribution.

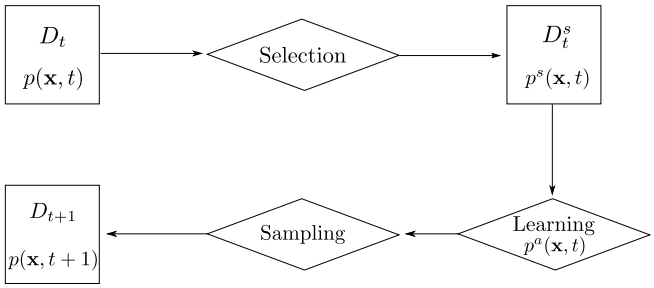


Fig. 1. Probability distributions determined by the components of an EDA. D_t, D_t^s, D_{t+1}: population and selected individuals at generation t and population at generation $t+1$; $p(\boldsymbol{x}, t)$, $p^s(\boldsymbol{x}, t)$, $p^a(\boldsymbol{x}, t)$: Probability distributions of the population,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

Algorithm 1: EDA scheme.

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

Algorithm 2: EDA with infinite populations, $\mathcal{A}$.

```
1 π ← ranking of the solutions x ∈ X given by an injective function f(x)
2 p_{t=0} ← generate the initial population
3 do {
4     Compute the probability of selection by means of φ as p_t^s = φ(p_t)
5     Compute p_t^a = M(p_t^s, π) to approximate p_t^s.
6     p_{t+1} ← p_t^a
7     t ← t + 1
8 } until Convergence
```

3. EDAs with infinite populations

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 [30], which is commonly used to theoretically analyze EAs. Regarding the field of EDAs, several works have assumed an infinite population in order to study and provide fundamental properties of this type of algorithms [14,19,25].

In EDAs with infinite populations, it is assumed that the empirical probability distribution induced by the solutions in D_t and D_t^s (Fig. 1) will converge to the underlying probability distributions $p(\boldsymbol{x}, t)$ and $p^s(\boldsymbol{x}, t)$ respectively, as the size of the sample tends to infinity. Therefore, $p(\boldsymbol{x}, t)$ and $p^s(\boldsymbol{x}, t)$ could be thought of as the population and the selected individuals at iteration t respectively [19]. In addition, it is assumed that the probability distribution $p(\boldsymbol{x}, t+1)$ of the next generation is equal to the distribution $p^a(\boldsymbol{x}, t)$ given by the probabilistic model. Therefore, the random errors caused by sampling finite sets of solutions from the probability distributions managed by the algorithm are canceled. This assumption does not limit the results. The taxonomy of problems is also valid with finite populations, as implemented in Algorithm 1. This assertion is explained more in detail in Section 5.3.

3.1. The algorithm and the population

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

The first step in the algorithm is to create the ranking π of the solutions $\boldsymbol{x} \in \mathcal{X}$. As commented above, an infinite population can be represented by the probability distribution $p(\boldsymbol{x}, t)$ that the algorithm manages at time t. Nevertheless, algorithm $\mathcal{A}$ does not explicitly work with these distributions $p(\boldsymbol{x}, t)$. Instead, at each generation, this algorithm manages a probability vector $\mathbf{p} = (p_1, p_2, \ldots, p_{|\mathcal{X}|})$ where each probability value p_i in $\mathbf{p}$ corresponds to the probability of the solution $\pi(i)$, i.e., $p(\pi(i)) = p_i$. The probability vector is associated to the ranking position instead of being associated to specific solutions of the search space. In this sense, it is always interpreted that the first value p_1 of the vector $\mathbf{p}$ is the probability

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

π_1	$\mathbf{p}_t$	$p(\boldsymbol{x}, t)$	$\mathbf{p}_t^s$
$(1,1)$	p_1	$p(\pi(1)) = p(1,1)$	p_1^s
$(0,1)$	p_2	$p(\pi(2)) = p(0,1)$	p_2^s
$(0,0)$	p_3	$p(\pi(3)) = p(0,0)$	p_3^s
$(1,0)$	p_4	$p(\pi(4)) = p(1,0)$	p_4^s
π_2	$\mathbf{p}_t$	$p(\boldsymbol{x}, t)$	$\mathbf{p}_t^s$
$(0,0)$	p_1	$p(\pi(1)) = p(0,0)$	p_1^s
$(1,0)$	p_2	$p(\pi(2)) = p(1,0)$	p_2^s
$(1,1)$	p_3	$p(\pi(3)) = p(1,1)$	p_3^s
$(0,1)$	p_4	$p(\pi(4)) = p(0,1)$	p_4^s

of the optimal solution, p_2 is the probability of the second best solution, and so on, with the last probability $p_{|\mathcal{X}|}$ corresponding to the solution with the worst 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 $\mathbf{p}_t = (p_1^t, p_2^t, \ldots, p_{|\mathcal{X}|}^t)$. Unlike the probability distribution $p(\boldsymbol{x})$, a vector $\mathbf{p}$ implies no specific assignment of probabilities to solutions. In order to obtain the probability distribution $p(\boldsymbol{x}, t)$ that algorithm $\mathcal{A}$ manages at time t, we use the ranking π. We have that $\pi^{-1}(\boldsymbol{x})$ gives the index of $\boldsymbol{x}$ in the ranking, so $p(\boldsymbol{x}, t)$ is the component $p_{\pi^{-1}(x)}$ of the vector $\mathbf{p}_t$.

The arrangement of the probability vector $\mathbf{p}$ and its relationship with the ranking π is illustrated in Table 2. In this example, we consider two different rankings π_1 and π_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 to the solution $(1,1)$ according to π_1. However, according to π_2, p_1 is associated to the solution $(0,0)$. Therefore, p_1 is the probability of the optimum in both cases, independently 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 π. Note that, since we have $2^n!$ different rankings π, 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:

$$\Omega_{|\mathcal{X}|} = \{(p_1, p_2, \ldots, p_{|\mathcal{X}|}) : \sum_{i=1}^{|\mathcal{X}|} p_i = 1, p_i \geq 0\}. \tag{2}$$

In addition, it is assumed that any initial vector $\mathbf{p}_0 = (p_1, \ldots, p_{|\mathcal{X}|})$ satisfies that $p_i < 1$ for all $i \in \{1, \ldots, |\mathcal{X}|\}$. 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.

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

$$\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots. \tag{3}$$

We use these sequences of probability vectors $\mathbf{p}$ to describe the behavior of the algorithm.

3.2. Selection

As indicated previously, we assume selection schemes based on the rank of the solutions within the population. Therefore, this class of selection only takes into account qualitative information about the function, i.e., it only uses the fact that $f(\boldsymbol{x}) > f(\boldsymbol{y})$, instead of the real value given by the function. We argue that many black-box algorithms use ranking information in the selection step to guide the search. In addition, ranking-based algorithms play an important role in the theory of black-box optimization. For instance, it has been shown in [31] that, for certain optimization problems, the use of ranking-based selection mechanisms facilitates the estimates of realistic time complexity results.

In algorithm $\mathcal{A}$, since any probability value p_i is interpreted as the probability of the solution $\pi(i)$, the selection can be simply defined by a function $\phi : \Omega_{|\mathcal{X}|} \to \Omega_{|\mathcal{X}|}$. This function modifies the probability values of the individuals according to the ranking positions in which they are. In this work, 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 in order to guarantee that the taxonomies of problems are valid for any implementation of ϕ.

- Property 1 (Impartiality). The selection operator is independent of the configuration $\boldsymbol{x}$ of a solution. This operator only takes into account the fitness value $f(\boldsymbol{x})$ 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 a rank-based selection scheme, given $\mathbf{p}$, we always obtain the same probability vector $\mathbf{p}^s = \phi(\mathbf{p})$ after selection, independently of π. 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 $\mathbf{p}^s = (p_1^s, \ldots, p_{|\mathcal{X}|}^s)$, 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.

In the context of EDAs, some examples of selection schemes implemented and studied under infinite population can be found in [14,19,25]. All these implementations satisfy Properties 1 and 2.

3.3. Approximation

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

The approximation $\mathcal{M}$ is the only operator in $\mathcal{A}$ that takes into account the specific $\boldsymbol{x}$ values of the variables (genotype) belonging to the problem solutions. Therefore, only $\mathcal{M}$ 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 $\mathcal{A}$ generates deterministic sequences of probability vectors when it is applied to solve a given problem. An iteration of this algorithm can be expressed by a function $\mathcal{G} : \Omega_{|\mathcal{X}|} \times \Pi \to \Omega_{|\mathcal{X}|}$ as $\mathbf{p}_{t+1} = \mathcal{G}(\mathbf{p}_t, \pi)$. 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 $\mathcal{G}$:

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

where $\mathcal{G}^2(\mathbf{p}_0, \pi) = \mathcal{G}(\mathcal{G}(\mathbf{p}_0, \pi), \pi)$ and then the following iterations are computed accordingly. The probability vector at iteration t is expressed as $\mathcal{G}^t(\mathbf{p}_0, \pi)$.

The convergence of EDAs has usually been studied by means of discrete dynamical systems defined on functions similar to $\mathcal{G}^t$. In [14,32], important insights and results about the convergence of some EDAs were provided using this approach.

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

Definition 1. Let π_1 and π_2 be the rankings 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 π_1 and π_2 are equivalent under $\mathcal{A}$, and by extension f_1 and f_2, if for any $\mathbf{p}_0 \in \Omega_{|\mathcal{X}|}$, $\mathcal{G}^t(\mathbf{p}_0, \pi_1) = \mathcal{G}^t(\mathbf{p}_0, \pi_2)$ for all $t = 1, 2, 3, \ldots$.

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 starting 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 $\boldsymbol{x}$ configurations. Therefore, if two functions are equivalent, we can say that the algorithm will have the same performance in terms of solving Eq. (1).

In Table 3, a very simple example with $n = 2$ binary variables illustrates the equality of vector sequences. Departing from the uniform distribution, algorithm $\mathcal{A}$ induces exactly the same sequence of probability vectors $\mathbf{p}_t$. If both sequences are equal from any initial $\mathbf{p}_0$, then we say that π_1 and π_2 are equivalent. Note, however, that the sequences of probability distributions $p(\boldsymbol{x}, t)$ are different.

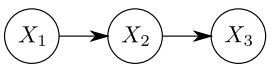
Definition 1 provides an equivalence relation because it is a reflexive, symmetric and transitive relation between functions. Given this equivalence relation, for each π, we have the corresponding equivalence class denoted by $[\pi]$. The equivalence relation partitions the space of functions into equivalence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

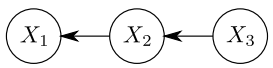
The equivalence between functions is defined by a given algorithm $\mathcal{A}$ which implements certain ϕ and $\mathcal{M}$. However, as discussed in the previous section, both operators do not play the same role. If two injective functions are equivalent under $\mathcal{A}$, then both functions will 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. Only the factorization used to approximate $p^s(\boldsymbol{x})$ can create different partitions of the space of problems. These partitions are independent of the implementation of ϕ.

Table 3
Two equal sequences when $\mathcal{A}$ is applied to π_1 and π_2. Example with $n = 2$ binary variables.

π_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	0.0038		0
(1, 0)	0.25	0.0625	0.0039	0.0001	…	0

π_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	0.0038		0
(0, 0)	0.25	0.0625	0.0039	0.0001	…	0





$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2) \qquad p(\boldsymbol{x}) = p(x_3)p(x_2|x_3)p(x_1|x_2)$$

$$p(\boldsymbol{x}) = \frac{p(x_1,x_2)p(x_2,x_3)}{p(x_2)}$$

Fig. 2. Example of factorizations given by Bayesian networks and the irreducible expression.

In [23], it was proved that all the injective functions are in the same equivalence class under an algorithm $\mathcal{A}$ when the implementation of $\mathcal{M}$ satisfies $p^a(\boldsymbol{x}, t) = p^s(\boldsymbol{x}, t)$. Therefore, this algorithm always behaves in the same way for any function.

5. Factorizations and the equivalence condition

The concepts and definitions about the taxonomy of problems that were abstractly discussed in the previous sections are studied here for EDAs whose approximation step is implemented by a factorization given by a Bayesian network. Different algorithms such as univariate marginal distribution algorithm (UMDA) [33], extended compact genetic algorithm (ECGA) [34] or factorized distribution algorithm (FDA) [25] use factorizations that can be expressed by a Bayesian network.

5.1. Factorizations given by Bayesian networks

A Bayesian network can be defined as a pair $(\mathcal{S}, \boldsymbol{\theta}_\mathcal{S})$ [6] where $\mathcal{S}$ is a directed acyclic graph (model structure) and $\boldsymbol{\theta}_\mathcal{S}$ is the set of parameters associated to the structure (model parameters). The structure $\mathcal{S}$ determines the set of conditional (in)dependences among the random variables of $\boldsymbol{X}$. According to the structure $\mathcal{S}$, the joint probability distribution $p(\boldsymbol{x})$ can be factorized by means of marginal and conditional probability functions.

In relation to algorithm $\mathcal{A}$, we consider that the function $\mathcal{M}$ approximates the distribution $p^s(\boldsymbol{x})$ by means of a new probability distribution $p^a(\boldsymbol{x})$ which factorizes according to the graph $\mathcal{S}$ as,

$$p^a(\boldsymbol{x}) = \prod_{i=1}^{n} p^s(x_i|\boldsymbol{pa}_i), \tag{4}$$

where $\boldsymbol{pa}_i$ denotes a value of the variables $\boldsymbol{Pa}_i$, the parent set of X_i in the graph $\mathcal{S}$. The conditional probability distributions can also be expressed as,

$$p(x_i|\boldsymbol{pa}_i) = \frac{p(x_i, \boldsymbol{pa}_i)}{p(\boldsymbol{pa}_i)}. \tag{5}$$

When the conditional probabilities of Eq. (4) are expressed as a division, some of the marginals belonging to the factorization can be simplified. This fact is illustrated in Fig. 2. In this example, we can see two different graphs and below them, the corresponding factorizations in terms of Eq. (4). At the bottom of the figure, we show the factorization when the conditional probabilities are expressed as a division and some marginals are simplified. This last factorization will be called irreducible. Of course, all the expressions presented in Fig. 2 are equivalent and therefore, they encode the same probability distributions. Nevertheless, the following results assume that the EDA implements the irreducible expression of Eq. (4).

Table 4
Ranking π given by an injective function with $n = 3$ variables.

π
(1, 1, 1)
(0, 1, 0)
(0, 0, 1)
(1, 0, 0)
(0, 1, 1)
(1, 0, 1)
(1, 1, 0)
(0, 0, 0)

5.2. Relationship between the objective function and the factorization

The marginal probability distributions of the factorization used to approximate the distribution of the selected individuals can be exactly calculated in algorithm $\mathcal{A}$. Then, given a set of indices K, the marginal probability function $p(\boldsymbol{x}_K)$ of $p(\boldsymbol{x})$ is defined as,

$$p(\boldsymbol{x}_K) = \sum_{\boldsymbol{x}\setminus\boldsymbol{x}_K} p(\boldsymbol{x}), \tag{6}$$

where $\boldsymbol{x}\setminus\boldsymbol{x}_K$ indicates that the sum of probabilities is over the different configurations of the variables that do not belong to $\boldsymbol{x}_K$.

The approximate probability distribution (Eq. (4)) is computed by calculating the marginal distributions as indicated in Eq. (6). By taking into account the function π, we can rewrite Eq. (6) as,

$$p(\boldsymbol{X}_K = \boldsymbol{x}_K) = \sum_{\boldsymbol{x}\setminus\boldsymbol{x}_K} p(\boldsymbol{x}) = \sum_{\tau\in Q_K^{\boldsymbol{x}_K}} p(\pi(\tau)), \tag{7}$$

where $Q_K^{\boldsymbol{x}_K} = \{\tau : \pi(\tau) = \boldsymbol{y} \wedge \boldsymbol{y}_K = \boldsymbol{x}_K\}$ is the set of ranking positions corresponding to the solutions $\boldsymbol{x}\in\mathcal{X}$, whose probabilities are used to calculate the marginal probability $p(\boldsymbol{X}_K = \boldsymbol{x}_K)$. In other words, $Q_K^{\boldsymbol{x}_K}$ is the set of ranking positions where the values of the variables given by K are equal to $\boldsymbol{x}_K$. The cardinality of a set $Q_K^{\boldsymbol{x}_K}$ is $2^{n-|K|}$. For each possible configuration $\boldsymbol{x}_K$ of the vector of variables $\boldsymbol{X}_K$, we have the corresponding set of ranking positions. Hence, we have $2^{|K|}$ different sets $Q_K^{\boldsymbol{x}_K}$ associated to the marginal distribution $p(\boldsymbol{x}_K)$. All these sets $Q_K^{\boldsymbol{x}_K}$ are arranged on a matrix $O_K = (o_{i,j})_{i=1,\ldots,2^{|K|},j=1,\ldots,2^{n-|K|}}$ which is associated to the marginal distribution $p(\boldsymbol{x}_K)$. We dispose a set $Q_K^{\boldsymbol{x}_K}$ at each row of the matrix O_K whose elements are arranged in the following way. The elements of the rows are sorted in ascending order $(o_{i,1} < o_{i,2} < \ldots < o_{i,2^{n-|K|}} \ldots)$ and the first column of the matrix is sorted in ascending order $(o_{1,j} < o_{2,j} < \ldots < o_{2^K,j})$. This type of matrices provides a unified description of the relationship between marginal distributions and objective functions.

In order to illustrate the matrix O_K associated to a marginal distribution $p(\boldsymbol{x}_K)$, we show an example based on the function of Table 4 and the marginal $p(x_1, x_2)$. According to this example, we have that $K = \{1, 2\}$ and the sets $Q_K^{\boldsymbol{x}_K}$ contain the following elements: $Q_{12}^{00} = \{3, 8\}$, $Q_{12}^{01} = \{2, 5\}$, $Q_{12}^{10} = \{4, 6\}$, $Q_{12}^{11} = \{1, 7\}$. The information related to the marginal $p(x_1, x_2)$ is arranged in the following matrix

$$O_{12} = \begin{pmatrix} 1 & 7 \\ 2 & 5 \\ 3 & 8 \\ 4 & 6 \end{pmatrix}.$$

We write $Q_{12}^{\boldsymbol{x}_K}$ and O_{12} instead of $Q_{\{1,2\}}^{\boldsymbol{x}_K}$ and $O_{\{1,2\}}$ in order to simplify the notation. The same is done in following examples.

As explained in the previous section, we consider the irreducible expression of Eq. (4) in order to approximate the distribution of the selected individuals. With this expression, the factorization contains different marginal distributions $p(\boldsymbol{x}_K)$ in the numerator and in the denominator. Nevertheless, we only need the numerator to express the relationship between the function and the factorization, since each term in the denominator is uniquely derived from the numerator (e.g., we have in Fig. 2 that $p(x_2) = \sum_{x_1} p(x_1, x_2) = \sum_{x_3} p(x_2, x_3)$).

In order to gather all the information needed to link the function and the factorization, we define the set $G_\pi = \{O_{K_i}\}_{K_i\in\mathcal{K}}$. The set $\mathcal{K}$ contains the subsets of indices corresponding to the factors of the numerator. We can have $|G_\pi| = m$ marginal distributions instead of n, as indicated in Eq. (4), because some terms of the factorization can be simplified when the conditional probabilities are expressed as a division.

The following example illustrates the set G_π. Given the function shown in Table 4 and the factorization

$$p(\boldsymbol{x}) = \frac{p(x_1, x_2)p(x_2, x_3)}{p(x_2)},$$

we have that $G_\pi = \{O_{12}, O_{23}\}$ and therefore,

$$G_\pi = \left\{ \begin{pmatrix} 1 & 7 \\ 2 & 5 \\ 3 & 8 \\ 4 & 6 \end{pmatrix}, \begin{pmatrix} 1 & 5 \\ 2 & 7 \\ 3 & 6 \\ 4 & 8 \end{pmatrix} \right\}.$$

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 Eq. (4). By using this condition, we carry out the partition of the space of functions into equivalence classes.

Theorem 1. *Let $\mathcal{A}$ be an EDA that implements $\mathcal{M}$ as $p^a(\boldsymbol{x}) = \prod_{i=1}^n p^s(x_i|\boldsymbol{pa}_i)$. Two functions π_1 and π_2 are equivalent under $\mathcal{A}$ if and only if the corresponding sets G_{π_1} and G_{π_2} are equal.*

Proof. Two functions π_1 and π_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 $\mathcal{G}^t(\mathbf{p}_0, \pi_1) = \mathcal{G}^t(\mathbf{p}_0, \pi_2)$ in every generation t for all initial $\mathbf{p}_0$ if and only if $G_{\pi_1} = G_{\pi_2}$.

Firstly, we show that if $G_{\pi_1} = G_{\pi_2}$, then $\mathcal{G}^t(\mathbf{p}_0, \pi_1) = \mathcal{G}^t(\mathbf{p}_0, \pi_2)$ for all t. Since the selection ϕ is independent of π, it is enough to prove that $\mathcal{M}(\mathbf{p}^s, \pi_1) = \mathcal{M}(\mathbf{p}^s, \pi_2)$ for any given $\mathbf{p}^s$. $G_{\pi_1} = G_{\pi_2}$ if and only if for any matrix $O_i^{\pi_1} \in G_{\pi_1}$, there exists a matrix $O_j^{\pi_2} \in G_{\pi_2}$ such that $O_i^{\pi_1} = O_j^{\pi_2}$ and vice versa. Therefore, according to Eq. (7), if $G_{\pi_1} = G_{\pi_2}$, 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 $\mathcal{G}^t(\mathbf{p}_0, \pi_1) = \mathcal{G}^t(\mathbf{p}_0, \pi_2)$ in every generation t for all initial $\mathbf{p}_0$, then $G_{\pi_1} = G_{\pi_2}$. 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 as follows. Let $\mathcal{K} = \{K_1, \ldots, K_m\}$ the set of the m subsets of index variables corresponding to factors of the numerator. For each sub-vector of variables $\boldsymbol{X}_{K_i}$, and for each possible configuration $\boldsymbol{x}_{K_i}$ of the variables $\boldsymbol{X}_{K_i}$, we consider an initial probability vector $\mathbf{p}_0 = (p_1, p_2, \ldots, p_{|\mathcal{X}|})$ such that $p_i > 0$ if $\pi_1[i] = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{X}_{K_i} = \boldsymbol{x}_{K_i}$, otherwise $p_i = 0$. These initial vectors have probability values greater than 0 only in the positions associated to solutions in which the sub-vector $\boldsymbol{X}_{K_i}$ is equal to $\boldsymbol{x}_{K_i}$ in π_1. Due to Property 2 of ϕ, we always obtain a vector $\mathbf{p}_0^s$ after selection with non-zero probabilities in the same positions as in $\mathbf{p}_0$. Then, we have that $p^s(\boldsymbol{X}_{K_i} = \boldsymbol{x}_{K_i}, t = 0) = 1$ for π_1 after selection. Since we know that $\mathcal{G}^1(\mathbf{p}_0, \pi_1) = \mathcal{G}^1(\mathbf{p}_0, \pi_2)$, it necessarily implies that there exists an assignment $\boldsymbol{y}_{K_i}$ for the variables $\boldsymbol{X}_{K_i}$ such that $p^s(\boldsymbol{X}_{K_i} = \boldsymbol{y}_{K_i}, t = 0) = 1$ for π_2. Therefore, the set $Q^{\boldsymbol{x}_{K_i}}$ associated to π_1 is equal to the set $Q^{\boldsymbol{y}_{K_i}}$ associated to π_2. Since this happens for all the possible assignments of the variables $\boldsymbol{X}_{K_i}$ and the algorithm generates the same sequence of probability vectors for all of the considered initial points, we have that $O_{K_i}^{\pi_1} = O_{K_i}^{\pi_2}$. The same argument is valid for all K_i in $\mathcal{K}$ and therefore, if the algorithm generates the same sequences of probability vectors from every initial point for π_1 and π_2, then $G_{\pi_1} = G_{\pi_2}$. □

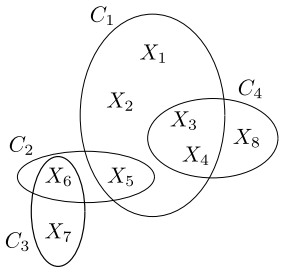
5.3. From infinite to finite populations

As previously mentioned, the equivalence between functions studied here is also valid for EDAs with finite population. Therefore, the theoretical results of this work can be translated to Algorithm 1, implementing ranking selection and approximation based on a Bayesian network. This fact can be explained as follows without using new mathematical machinery. If two functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are equivalent under $\mathcal{A}$, then, despite the random sampling, we can ensure that for any sequence of populations $D_1, \ldots, D_t$ generated by Algorithm 1 for $f_1(\boldsymbol{x})$ there exists a sequence of population $D'_1, \ldots, D'_t$ for $f_2(\boldsymbol{x})$ with exactly the sample probability. Basically, if we mathematically model the application of the EDA to a function with finite population as a Markov chain, where the states of the Markov chain are the different populations, then, when two functions f_1 and f_2 are equivalent under our framework, there exists a re-order of the states of the Markov chain of f_2 in such a way that the transition matrices are exactly the same.

In essence, two equivalent functions are equal from the point of view of Algorithm 1. This fact does not imply that the algorithm will produce the same runs for both functions at the same time deterministically. It implies that the overall performance of the algorithm, taking into account all possible runs, is essentially the same for both functions. We can say that the EDA will exhibit the same behavior for both functions. In this sense, the results obtained by using an infinite population EDA model are valid for finite samples.

6. Description of the equivalence classes. Case study: chordal Markov networks

In this section, we specifically show the changes or transformations that can be made on a function π in order to generate all the functions of the class $[\pi]$. By synthesizing the essential transformations, we can describe and count the members of a class. Considering general Bayesian networks for this purpose entails an unapproachable casuistry due to the use of directed graphs. Therefore, with the aim of providing an illustrative and concise description of the equivalence classes in this part of the paper, we assume that the EDA implements a chordal Markov network. We take into account this type



$$p(\boldsymbol{x}) = p(x_1, x_2, x_3, x_4, x_5)p(x_6|x_5)p(x_7|x_6)p(x_8|x_3, x_4)$$

Fig. 3. Graphical representation of a junction tree and the corresponding factorization.

of models because, on the one hand, they are widely used in EDAs [26] and, on the other hand, they provide a scenario of bounded complexity in which we can explicitly show how to count and describe equivalent functions.

One of the aspects that influences the behavior of EDAs that use chordal Markov networks is the number of overlaps between the factors of the factorizations determined by the models. The analysis made in this section helps to reveal how the type and number of overlaps are reflected in the number of equivalent functions that can be obtained with these models.

6.1. Chordal Markov networks

Markov networks [24,35] are probabilistic graphical models where the factorization of the joint probability distribution is encoded by an undirected graph. The application of this class of probabilistic models in EDAs is a current research line [26]. Chordal Markov networks are used in FDA [25,36], one of the early multivariate EDAs, and in more recent algorithms [37]. This type of networks also constitutes the graphical representation of the exact factorizations [25] that can be associated to additive decomposable functions [38].

The following definitions are needed in order to formally introduce the factorizations involved in the analysis. Given a graph $G = (V, E)$, a clique C in G is a fully connected subset of V. C is maximal when it is not contained in any other clique. From now on, any mentioned clique C is assumed to be a maximal clique. A graph G is said to be chordal if every cycle of length four or more has a chord. Note that the number of maximal cliques is equal to or less than $|V| = n$.

In order to obtain a factorization of the probability function $p(\boldsymbol{x})$ according to a chordal graph G, we order the cliques of the graph according to the running intersection property [24]. Let $C_1, \ldots, C_m$ be a sequence of the cliques of the graph. Let

$$S_i = C_i \cap \{C_1 \cup \ldots \cup C_{i-1}\}, i = 2, \ldots, m$$

be the separator sets. Since $S_i \subset C_i$, the residual sets are defined as

$$R_i = C_i \backslash S_i.$$

Then, the sequence of cliques $C_1, \ldots, C_m$ satisfies the running intersection property if the following condition is fulfilled [35]:

for all $i > 1$ there is a $j < i$ such that $S_i \subset C_j$.

The running intersection property guarantees that the separator sets S_i separate the residual sets R_i from the sets $\{C_1 \cup \ldots \cup C_{i-1}\} \backslash S_i$. This fact can be graphically represented by means of a junction tree, associated to the chordal graph. Each node of a junction tree corresponds to a clique such that any two cliques containing a node X_i are either adjacent in the junction tree, or connected by a chain made entirely of cliques that contain X_i.

A Markov network [24] can be defined as a pair (G, Ψ) where G is an undirected graph and $\Psi = \{\psi_1(\boldsymbol{x}_{C_1}), \ldots \psi_m(\boldsymbol{x}_{C_m})\}$ is a set of potential functions defined on the cliques $\{C_1, \ldots, C_m\}$ of G that define a probability function $p(\boldsymbol{x})$ by means of

$$p(\boldsymbol{x}) = \prod_{i=1}^{m} \psi_i(\boldsymbol{x}_{C_i}) \tag{8}$$

If the graph G is chordal, then $p(\boldsymbol{x})$ can also be factorized as follows

$$p(\boldsymbol{x}) = \prod_{i=1}^{m} p(\boldsymbol{x}_{R_i}|\boldsymbol{x}_{S_i}) \tag{9}$$

where R_i and S_i are the residuals and separators of the cliques defined above. In this case, the graphical representation of the factorization can be given by a junction tree. In Fig. 3, we show an example of a junction tree and the corresponding factorization. Note that these factorizations can also be expressed by means of Bayesian networks.

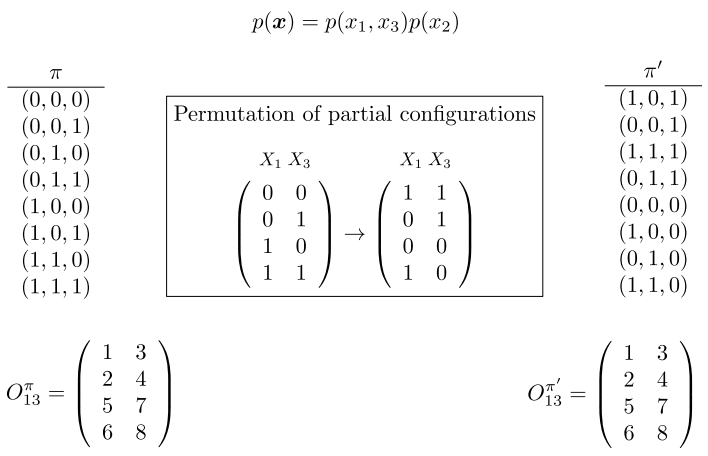


Fig. 4. Generating an equivalent function by means of permutations of partial configurations, example with $n = 3$.

In order to describe and count equivalent functions in the next section, we make the following assumption in the factorizations. No separator set S_i is a proper subset of another separator S_j. This type of factorizations can be formulated as

$$p(\boldsymbol{x}) = \frac{\prod_{i=1}^{m} p(\boldsymbol{x}_{C_i})}{\prod_{i=1}^{m-1} p(\boldsymbol{x}_{S_i})}, \tag{10}$$

This assumption in the factorization allows us to describe and count equivalent functions in a scenario of reasonably bounded casuistry. Since we assume that the factorization is given by a junction tree, we do not consider disconnected cliques. Nevertheless, the application of the following results to a forest of junction trees is straightforward.

6.2. Generating equivalent functions

The equivalent functions can be generated by combining certain transformations of the solutions $\boldsymbol{x}$ arranged in the ranking π. In order to do that, we firstly introduce the matrices of partial configurations associated to a clique. These matrices are useful to describe and count the modifications that we can carry out in a function to generate equivalent functions. A matrix of partial configurations associated to a clique C_i is simply a matrix where each row contains a configuration of the variables $\boldsymbol{X}_{C_i}$. Therefore, this matrix has $2^{|C_i|}$ rows and $|C_i|$ columns (binary case). The rows of this matrix do not have any predefined order. We consider that any permutation of the rows of this matrix generates a new function when this permutation of partial configurations is translated into the function.

The previous concepts are illustrated in Fig. 4 with a function defined on three variables. In this example, we focus only on the marginal distribution $p(x_1, x_3)$ of the factorization indicated at the top of the picture. We have the ranking π on the left and the equivalent ranking π' on the right. In the center of the figure, we show a permutation of partial configurations corresponding to the variables X_1 and X_3 that produces the equivalent ranking π'. According to this permutation, the partial configuration $(X_1 = 0, X_3 = 0)$ becomes $(X_1 = 1, X_3 = 1)$. Therefore, this change is translated into the function by replacing all the positions of the ranking in which $X_1 = 0$ and $X_3 = 0$ by $X_1 = 1$ and $X_3 = 1$ respectively. In general, if we change the partial configuration $\boldsymbol{x}_C$ by $\boldsymbol{y}_C$, we assume that this change is carried out in all the $2^{n-|C|}$ positions of the function that contain the configuration $\boldsymbol{x}_C$ and that a new function has been generated.

Below the functions π and π' in Fig. 4, we have the corresponding matrices of rankings associated to the marginal $p(x_1, x_3)$. It is easy to check that this permutation of the partial configurations generates an equivalent function because the matrix O_{13} associated to $p(x_1, x_3)$ is the same.

If there are no overlapping variables in the factorization, any permutation of the rows of a matrix of partial configurations produces an equivalent function. On the contrary, when the factorization has overlapping variables, not all the permutations are allowed. Both scenarios are discussed next.

6.2.1. Factorizations without overlapping

This type of factorizations, usually called marginal product factorizations, can be formulated as follows

$$p(\boldsymbol{x}) = \prod_{i=1}^{m} p(\boldsymbol{x}_{C_i}), \tag{11}$$

and a possible graphical representation is shown in Fig. 5. For instance, this type of factorizations is used by the ECGA [34].

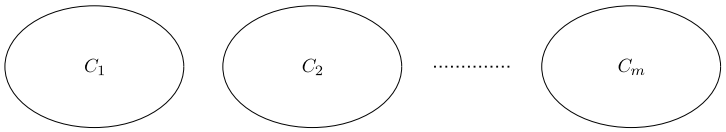


Fig. 5. Graphical representation of a factorization without overlapping variables.

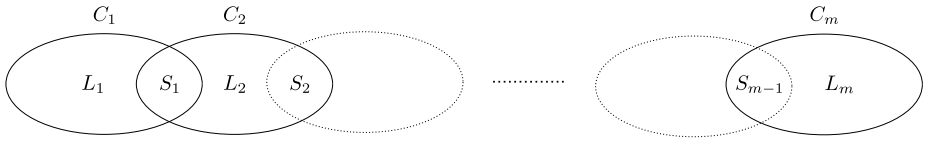


Fig. 6. Graphical representation of a factorization given by a chain of cliques.

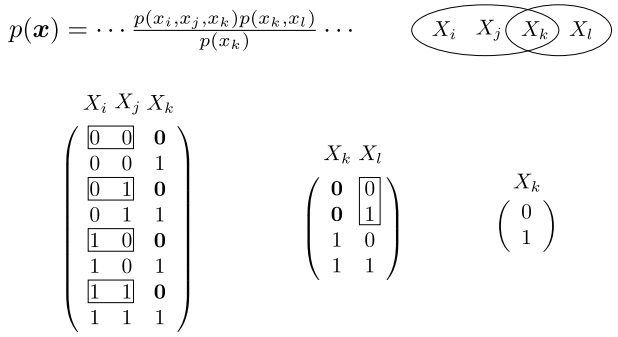


Fig. 7. Permutations of partial configurations when the factorization has overlapping variables.

We consider a matrix of partial configurations associated to each clique C_i. Since there are no overlaps, the possible configurations of the variables belonging to each clique can be freely permuted. Therefore, we can generate $2^{|C_i|}!$ equivalent functions for each clique. As we have m independent cliques, we can generate

$$\prod_{i=1}^{m} 2^{|C_i|}! \tag{12}$$

equivalent functions.

In addition, there is another type of transformation that can be made in a function in order to generate equivalent functions. This type of change is related to the swapping or permutation of variables introduced in [23]. Since we are dealing with disjoint cliques, we can permute, among the cliques of the same size, the complete sets of variables belonging to them in order to create new equivalent functions.

By assuming that all the cliques are of the same size $|C|$ for the sake of simplicity, we can conduct $m!$ permutations of the sets of variables belonging to each clique and, hence, the final number of functions per class in this case is

$$m!2^{|C|}!^m. \tag{13}$$

6.2.2. Factorizations with overlappings

In this section, we deal with factorizations of the form of Eq. (10), which are given by a chordal Markov network where no separator set is a proper subset of another separator. We start by considering a chain of cliques whose graphical representation is shown in Fig. 6. In each clique, we distinguish between sets L_i of free variables and sets S_i of separators. As can be seen in Fig. 6, we have that $C_1 = L_1 \cup S_1$, $C_2 = L_2 \cup S_1 \cup S_2$ and so on until $C_m = L_m \cup S_{m-1}$.

In this scenario, we also take into account the matrices of partial configurations associated to the variables of each clique. However, in this case, not all the permutations of partial configurations can generate equivalent functions. The permutations of partial configurations must be conditioned to the values of the separators in order to keep the matrices of rankings intact. This restriction is illustrated with the example of Fig. 7. In this example, we only consider the part of the factorization indicated at the top of the figure. The graphical representation, on the right of the factorization, clearly indicates that X_i and X_j are the free variables of the first clique, X_l is the free variable of the second clique and X_k is the separator. Below the factorization and its graphical representation, we show the corresponding matrices of partial solutions. We consider a matrix per each marginal in the factorization, including the marginal associated to the separator. In the matrices associated to the marginals of the numerator, we have written in bold the value 0 of the variable X_k. This indicates that the permutations of the partial configurations are conditioned to the value of X_k. Thus, we can permute, on the one

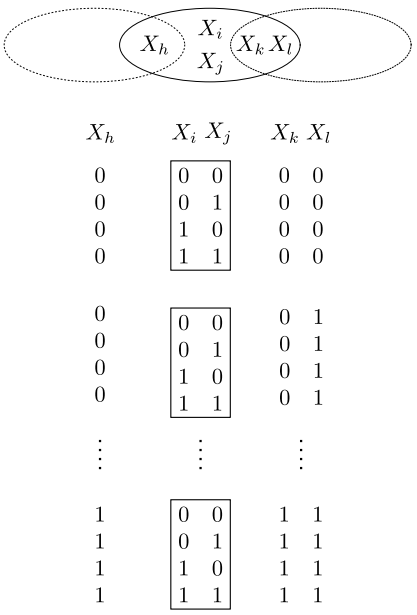


Fig. 8. Permutations of the free variables for a clique within the chain.

hand, the values of the free variables when $X_k = 0$ (these configurations of the free variables are within a square). On the other hand, and independently, we can permute the values of the free variables when $X_k = 1$. In no case can we permute partial configurations in which X_k takes a different value. The partial configurations of the matrix associated to a separator can be permuted without restrictions in order to generate equivalent functions. This type of movements guarantees that the associated matrices of rankings remain intact.

The two kinds of variables presented in this section (free and separators) are used to create equivalent functions in an independent way. On the one hand, the partial configurations associated to the separators can be permuted without restrictions. A set of variables S_i has $2^{|S_i|}$ partial configurations and, hence, we can conduct $2^{|S_i|}!$ permutations of these configurations to create equivalent functions. In the case of having a chain of cliques as shown in Fig. 6, we can generate

$$\prod_{i=1}^{m-1} 2^{|S_i|}! \tag{14}$$

equivalent functions considering only the separators. On the other hand, we have the equivalent functions that can be generated from the free variables. In this case, the permutations of the partial configurations associated to the set of variables L_i are conditioned to the values of the separators, as seen in the example of Fig. 7. Thus, we permute the partial configurations of the free variables $\boldsymbol{X}_{L_i}$ in the clique C_i given a specific configuration $\boldsymbol{X}_{S_i} = \boldsymbol{x}_{S_i}$ of the variables belonging to a separator. The partial configurations associated to the set L_i can be independently permuted for each assignment of the variables of the separator set. In the factorization given by a chain of cliques (see Fig. 6), only the first and last cliques have an overlap, whereas the remaining cliques have two overlappings. In the case of having an overlap, we can carry out $2^{|L|}!$ permutations of the partial configurations associated to the free variables for each configuration of the overlapping variables $\boldsymbol{X}_S$. Since the permutations conditioned to different configurations of $\boldsymbol{X}_S$ can be made independently, we can generate

$$2^{|L_1|}!^{2^{|S_1|}} \tag{15}$$

equivalent functions associated to the first clique and

$$2^{|L_m|}!^{2^{|S_{m-1}|}} \tag{16}$$

equivalent functions associated to the last clique.

When we have two separator sets S_{i-1} and S_i associated to a free set L_i, we have to consider the partial configurations associated to the set $S_{i-1} \cup S_i$ to condition the permutations corresponding to the free variables. This scenario is illustrated in Fig. 8. At the top of the figure we have a graphical representation of a clique which is somewhere within the chain of cliques that forms the factorization. It can be seen that X_i and X_j are free variables, X_h is the overlapped variable on the left and X_k, X_l are the overlapping variables on the right. In general, the configurations of the free variables can be permuted independently for each assignment of the overlapping variables. Taking into account the separator sets S_{i-1} and S_i, we

have $2^{|S_{i-1}|+|S_i|}$ possible configurations of the corresponding variables. Since we can carry out $2^{|L_i|}!$ permutations of the configurations of the free variables for each assignment of the overlapping variables, we can generate

$$2^{|L_i|}!2^{|S_{i-1}|+|S_i|}$$

equivalent functions associated to a set L_i. Therefore, we have

$$\prod_{i=2}^{m-1} 2^{|L_i|}!2^{|S_{i-1}|+|S_i|} \tag{17}$$

equivalent functions associated to intermediate cliques.

In summary, the total number of equivalent functions that we can generate by means of free and overlapping variables in a chain of cliques is equal to the product of Eq. (14), Eq. (15), Eq. (16) and Eq. (17).

At this point, we can generalize the number of overlappings per clique. In the general case, we can have q sets $S^1 \dots S^q$ of overlapping variables in a clique C. Following the aforementioned arguments, we can conduct $2^{|L|}!$ permutations of the configurations of the free variables for each assignment of the complete set $\bigcup_{k=1}^{q} S^k$ of overlapping variables in a clique C. Therefore, we can generate

$$2^{|L_i|}!2^{\sum_{j=1}^{q_i} |S_i^j|}$$

equivalent functions based on the free variables of a clique C_i.

Finally, given a junction tree, in which no separator set is a proper subset of another separator set, and taking into account the operations mentioned so far, we can generate

$$\prod_{i=1}^{m} 2^{|L_i|}!2^{\sum_{j=1}^{q_i} |S_i^j|} \prod_{k=1}^{v} 2^{|S_k|}! \tag{18}$$

equivalent functions. In this expression, we consider m sets of free variables and v sets of overlapping variables in the factorization. Note that we could have more than two cliques separated by the same set of variables. If all the separators are different, then $v = m - 1$.

Until now, we have generated equivalent functions taking into account permutations of partial configurations in the function according to the free and overlapping variables of the factorization. Nevertheless, as in the analysis of the factorizations without overlappings, we cannot describe all the equivalent functions belonging to a class by means of these permutations of partial configurations. To complete the description, we need to consider the permutation of variables.

In the previous section we showed that, if the factorization is given by disjoint cliques of the same size, it is straightforward to calculate the equivalent functions associated to the permutation of variables (or cliques in a junction tree). However, when we have cliques of different sizes or, as in the current case, overlappings between cliques, not all the permutations of variables are allowed and this becomes a non-trivial problem. The description of this type of equivalences can be formalized by using the concept of graph automorphism or, in our particular case, the automorphisms of junction trees. Analyzing equivalent functions by counting and describing the automorphisms of a junction tree is a complex task which is beyond the scope of this paper, nevertheless, we illustrate this type of equivalences by means of a simple examples. For the first example, we assume a chain of cliques of the same size (Fig. 6 can illustrate this model). In this case, the only allowed permutation of the variables of the cliques is to reverse their order, i.e., the variables belonging to C_1 are swapped with the variables of C_n, the variables of C_2 are swapped with the variables of C_{n-1} and so on. Therefore, the total number of equivalent functions for this type of model is equal to the number given by Eq. (18) (with $v = m - 1$) times 2.

7. Factorizations and neighborhood systems

The relationship between EDAs and the local optima of the function has been previously studied by means of dynamical systems in univariate [14,32] and bivariate implementations [14]. These studies are significant to better understand the convergence behavior of the EDA and its predisposition to escape or get trapped in local optima. In [23], it was shown that this relationship can also be studied, at least for univariate EDAs, by means of the equivalence classes that this type of algorithm can create in the space of functions.

In this section, we extend the results presented in [23] to EDAs that can implement a factorization of the form given by Eq. (10), which is based on the chordal Markov networks introduced in the previous section. In order to do that, we create a neighborhood system in the search space $\mathcal{X}$ associated to Eq. (10). The distance between two solutions $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ is defined as the number of cliques C_i where the configurations $\boldsymbol{x}_{C_i}$ and $\boldsymbol{y}_{C_i}$ of the corresponding variables are different, minus the number of overlappings S_i where the configurations $\boldsymbol{x}_{S_i}$ and $\boldsymbol{y}_{S_i}$ are different. This can be formulated as

$$D_p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} d(\boldsymbol{x}_{C_i}, \boldsymbol{y}_{C_i}) - \sum_{i=1}^{m-1} d(\boldsymbol{x}_{S_i}, \boldsymbol{y}_{S_i}) \tag{19}$$

where

$$d(\boldsymbol{x}_K, \boldsymbol{y}_K) = \begin{cases} 0 & \text{if } \boldsymbol{x}_K = \boldsymbol{y}_K \\ 1 & \text{otherwise} \end{cases}$$

The distance presented in Eq. (19), associated to a factorization, verifies the following property.

Theorem 2. *Given a factorization of the form of Eq. (10), any function π' belonging to the class $[\pi]$ verifies that $D_p(\pi(i), \pi(j)) = D_p(\pi'(i), \pi'(j))$ for all $i, j \in \{1, \ldots, |\mathcal{X}|\}$.*

Proof. Since two equivalent functions π and π' have the same set G_π, then the following is verified. For any pair of ranking positions i, j, if they are in the same Q_C, then we have the same partial configuration $\boldsymbol{x}_C$ at positions i and j, both in π and π'. Therefore, $d(\boldsymbol{x}_C, \boldsymbol{x}_C) = 0$ in both functions. On the contrary, if i and j are not together in Q_C, then we have two different partial configurations $\boldsymbol{x}_C$ and $\boldsymbol{y}_C$ at positions i and j respectively, both in π and π'. Therefore, $d(\boldsymbol{x}_C, \boldsymbol{y}_C) = 1$ in both functions. The same argument is valid for every clique and for all overlapped sets S_i, although they are not considered in G_π. Since Eq. (19) takes into account all the cliques and overlapped sets in the factorization, if two functions π and π' have the same set G_π, then $D_p(\pi(i), \pi(j)) = D_p(\pi'(i), \pi'(j))$ for all $i, j \in \{1, \ldots, |\mathcal{X}|\}$. Since all the functions in the same class have the same set G_π, any function $\pi' \in [\pi]$ verifies that $D_p(\pi(i), \pi(j)) = D_p(\pi'(i), \pi'(j))$ for all i, j. □

This means that if we define a distance matrix between all the solutions, the distance matrix is the same for all the equivalent functions.

Given a factorization, we define the neighbors of a solution $\boldsymbol{x}$ as those solutions $\boldsymbol{y}$ such that $D_p(\boldsymbol{x}, \boldsymbol{y}) = 1$. The relation between the equivalence classes and the local optima, according to the neighborhood created by the solutions at distance 1, can be deduced from Theorem 2 and it is formally established by the following corollary.

Corollary 1. *Given a factorization of the form of Eq. (10) and the associated distance $D_p(\boldsymbol{x}, \boldsymbol{y})$, all the functions in the same equivalence class $[\pi]$ have the same number of local optima and in the same ranking positions.*

Proof. A local optimum can be defined according to the position that it holds in the ranking π and the distance to the preceding solutions. Then, given a factorization, a solution $\pi(i) = \boldsymbol{x}$ is a local optima for π if $D_p(\pi(i), \pi(j)) > 1$ for all $j < i$. By Theorem 2, we know that any function $\pi' \in [\pi]$ verifies that $D_p(\pi(i), \pi(j)) = D_p(\pi'(i), \pi'(j))$ for all i, j. Therefore, if there is a local optima in the position i of π, then there is a local optima in the same position i of π' and vice versa and, hence, the functions of a class always have the same number of local optima and in the same ranking positions. □

Besides these properties, the distance presented in Eq. (19) also satisfies another important property related with the convergence of the EDA. The following result can be directly deduced from Theorem 3 and Theorem 7 presented in [14]. The nomenclature has been adapted to the current work for the sake of simplicity.

Corollary 2. *Let $\mathcal{A}$ be an EDA that implements tournament selection and the approximation step according to Eq. (10). If the solution $\boldsymbol{y}$ is not a local optimum, i.e. there exists a solution $\boldsymbol{z}$ such that $D_p(\boldsymbol{y}, \boldsymbol{z}) = 1$ and $f(\boldsymbol{z}) > f(\boldsymbol{y})$, the algorithm cannot converge to the point $\mathbf{p} = (0, \ldots, p_i = 1, \ldots, 0) \in \Omega_{|\mathcal{X}|}$ where $i \in \{1, \ldots, 2^n\}$ is the position of the solution $\boldsymbol{y}$ in ranking π.*

Roughly speaking, it means that algorithm $\mathcal{A}$ cannot converge to a solution which is not a local optimum according to the distance given by Eq. (19). It is not needed to prove this result because it is only an adaptation of the results presented in [14]. A more exhaustive analysis of the properties of this distance is open for further research. Moreover, alternative distances related to factorizations could be developed.

Note that if we only consider the Hamming distance $H(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|$ and multivariate EDAs, we can have functions with different numbers of local optima in the same equivalence class. Therefore, for non-univariate EDAs, some of the local optima (according to Hamming distance) do not have any kind of special influence on the behavior of the algorithm as all the functions belonging to a class have the same behavior.

In order to illustrate the neighborhood systems associated to factorizations, we show an example with $n = 3$ in Fig. 9. It presents three neighborhood systems corresponding to the three factorizations indicated in the label of the figure. The solutions are disposed in an octagon in order to provide a clear graphical representation of the neighborhood systems. Two solutions connected by a line in the octagon are neighbors, i.e., they are at distance 1. The distance between two solutions is the minimum number of edges needed to move from one solution to the other. Fig. 9(a) represents the Hamming distance, whereas Figs. 9(b) and 9(c) show the distances among the solutions for the corresponding factorizations. The factorization associated to Fig. 9(b) contains the marginal distribution $p(x_1, x_2)$ that links the variables X_1 and X_2 as if they were a single variable of four states. Then, the distance between any pair of configurations of the variables X_1 and X_2 is always equal to 1. In the neighborhood system presented in Fig. 9(c), besides linking the variables X_1 and X_2, the marginal distribution $p(x_2, x_3)$ also links the variables X_2 and X_3. Therefore, the distance between any pair of configurations of these two last variables is equal to 1. By using Eq. (19), we can calculate the distances between every pair of solutions. Different types of

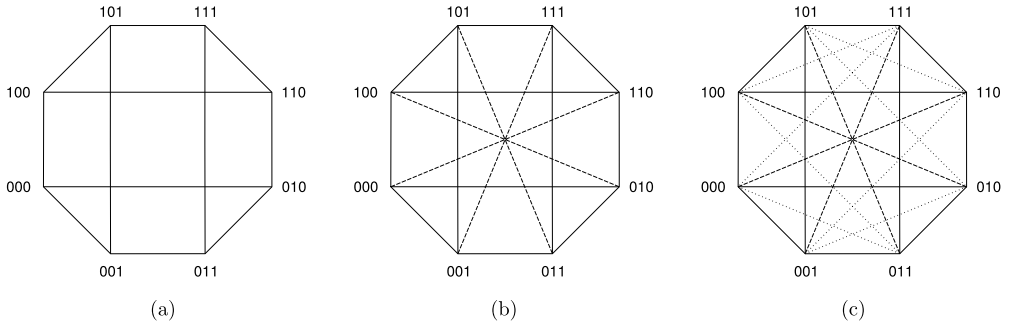


Fig. 9. Neighborhood systems associated to the following factorizations of three variables: (a) $p(\boldsymbol{x}) = p(x_1)p(x_2)p(x_3)$, (b) $p(\boldsymbol{x}) = p(x_1, x_2)p(x_3)$, (c) $p(\boldsymbol{x}) = p(x_1, x_2)p(x_2, x_3)/p(x_2)$.

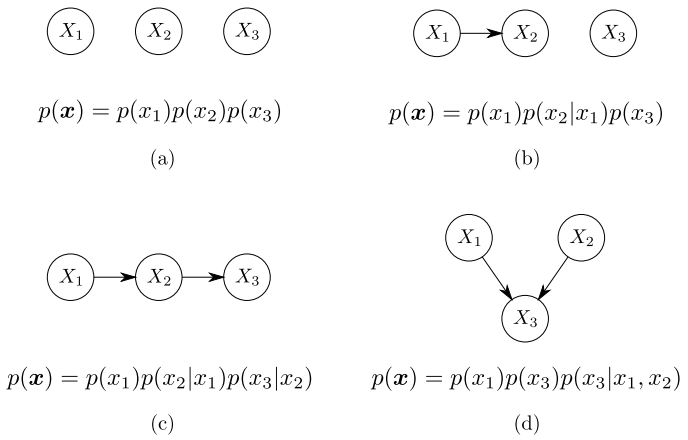


Fig. 10. Factorizations used to implement the different EDAs. (a) Univariate EDA. (b) Arc EDA. (c) Chain EDA. (d) V EDA.

lines are used in order to clearly distinguish the new neighbors that are generated as the complexity of the factorization increases. Thus, we use continuous lines to represent the neighborhood system for the factorization $p(\boldsymbol{x}) = p(x_1)p(x_2)p(x_3)$. We use dashed lines to represent the new neighbors generated when the factorization $p(\boldsymbol{x}) = p(x_1, x_2)p(x_3)$ is considered and dotted lines to represent the new neighbors associated to the factorization $p(\boldsymbol{x}) = p(x_1, x_2)p(x_2, x_3)/p(x_2)$. Note that, in this case, the neighborhood system of the simpler factorization is contained in the system of the more complex one.

8. Numerical experiments in $S = \{0, 1\}^3$

In this section, taking into account the theoretical tools developed in the previous sections, we carry out an exhaustive analysis of the different behaviors that EDAs implementing a Bayesian network can exhibit for the injective functions in $\{0, 1\}^3$. Since we assume infinite populations, the runs of the EDA are numerical simulations of algorithm $\mathcal{A}$. In particular, 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, ii) the probability of the optimum along the generations. We consider that the size of the basin of attraction of the optimum $\boldsymbol{x}^*$ is the number of initial points that converge to $\boldsymbol{x}^*$ after conducting a predefined number of executions of the algorithm. We assume that the smaller the size of the basin of attraction in a class, the more difficult the problems in that class are. Moreover, when two classes have the same basin size, then the longer the 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 behavior, we will put the previous complexity results in relation to this problem characteristic.

8.1. Experimental design

We take into account all the possible π that can be generated over the search space $\mathcal{X} = \{0, 1\}^3$. Therefore, we consider $2^3! = 40\,320$ functions.

Depending on the factorization used by the EDA, we will obtain a different partition of the space of problems. In order to show all possible behaviors that an EDA can exhibit when it uses a Bayesian network, we only need to implement four different factorizations. The graphs and the corresponding factorizations are presented in Fig. 10 (the remaining factorizations are equivalent to one of the four described here). By using Theorem 1, it is possible to group the functions and calculate

the number of classes for each factorization. In particular, we have: 840 classes when the EDA implements a univariate model (Fig. 10(a)), 840 classes when an arc is added (Fig. 10(b)), 630 classes when a chain model is used (Fig. 10(c)) and 315 classes when the algorithm implements a so-called V-structure model (Fig. 10(d)). For the experiments, we only need to consider one function per class because the algorithm behaves equivalently for all the functions in the same class. 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 (factorizations), 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. For each function, we launch 10 000 EDA runs, one from each previously generated initial probability vector.

We use two-tournament selection according to [14] 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. Since we deal with injective functions, two solutions cannot have the same function value. In algorithm $\mathcal{A}$, the probability vector $\mathbf{p}^s$ after tournament selection can be computed from the vector $\mathbf{p}$ as follows:

$$p_i^s = p_i^2 + 2p_i \sum_{j=i+1}^{n} p_j. \tag{20}$$

Note that this formulation obeys the properties that we imposed to ϕ in Section 3.

The stopping condition of the algorithm is a maximum of 30 iterations. This number of generations provides a satisfactory trade-off between accuracy in the numerical results and computational cost. Since the algorithm does not always converge exactly to 1 in 30 iterations, we consider that the algorithm has converged when the probability assigned to a solution is greater than $1 - 10^{-15}$.

In the numerical analysis, the size of the basins of attraction is 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.

8.2. Results

As previously discussed, the equivalence classes have a strong connection with neighborhood systems on $\mathcal{X}$ and this fact can play an important role in the numerical analysis. In this section, the relationship between EDAs and local optima can only be studied for the Univariate EDA, the Arc EDA and the Chain EDA because Eq. (19) is not valid for the V-structure factorization. Remember that in Fig. 9(a), (b) and (c) we can find the neighborhood systems associated to the first three EDAs respectively. In this section, we distinguish between local optima and global optimum. Thus, when we talk about local optima we are not considering the global optimum.

According to the numerical simulations, all the local optima predicted by the distance associated to the corresponding factorization have a basin of attraction greater than zero. This suggests that the local optima indicated by Eq. (19) can be associated to attractive fixed points. However, to the best of our knowledge, there is a lack of theoretical results regarding the convergence of EDAs that implement general factorizations and regarding their relationship with the corresponding neighborhood systems. Therefore, the simulations presented in this section could provide valuable information and intuition to further develop theoretical results related with this issue.

8.2.1. Analysis based on basins of attraction

The basins of attraction of the Univariate EDA were already presented in [23]. Nonetheless, we show those results here (Fig. 11) in order to provide a complete perspective of the behavior of the EDA.

In this analysis, we show the basins of attraction of the optimum for each class by means of different colors. The sizes of these basins of attraction are interpreted in terms of problem difficulty.

Each of the EDAs that we consider generates a different partition of the space of problems. Besides the Univariate EDA, we show the basins of attraction for the Arc EDA and the Chain EDA in Fig. 12 and Fig. 13 respectively (the squares within the plots represent the equivalence classes). We do not show the corresponding results for the V EDA because it always reaches the optimum in all the classes. Nevertheless, we will provide a brief discussion about this algorithm.

The color bar on the right of these figures indicates the relation between the colors and the size of the basin. At the top of the spectrum, the green color is assigned to the largest basins of attraction which indicates easy problems. At the bottom, the red and darker colors represent small basins of attraction and hence, they reveal the hardest problems.

In addition, the classes have been grouped by the number of local optima. Thus, Figs. 11, 12 and 13 have been divided into different parts, separated by vertical dashed lines. In each of these parts, the classes are ordered according to the size of the basins of attraction. We place the classes without local optima on the left and the classes with the maximum number of local optima on the right. In the case of the Univariate EDA, we can find up to three local optima (apart from the global optimum) in a function. For the Arc EDA and the Chain EDA, we can only find one local optimum.

For the Univariate EDA and the Arc EDA, the green classes cover all the problems without local optima. However, in the case of the Chain EDA, we can note that some classes without local optima take yellow color because some runs of

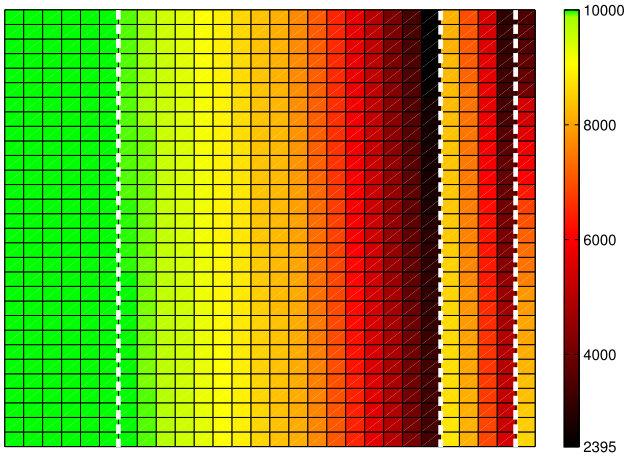


Fig. 11. Univariate EDA. Size of the basin of attraction of the optimal solution for each class (840 classes). The image has been divided into four parts separated by vertical dashed lines. From left to right, we have the classes with zero, one, two and three local optima respectively. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

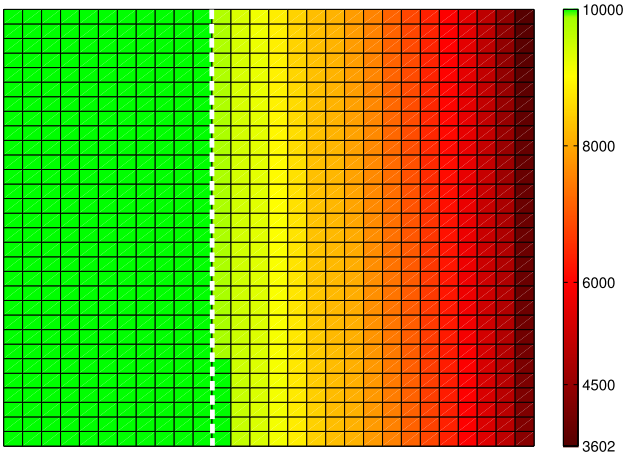


Fig. 12. Arc EDA. Size of the basin of attraction of the optimal solution for each class (840 classes). The image has been divided into two parts separated by a vertical dashed line. On the left we have the classes without local optima and, on the right, we have the classes with one local optimum. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

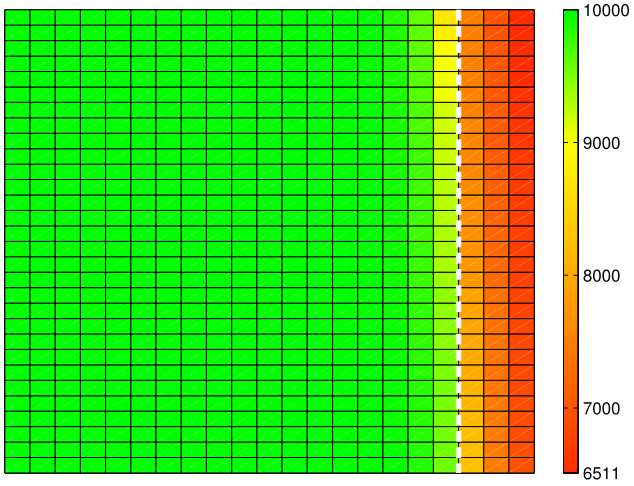


Fig. 13. Chain EDA. Size of the basin of attraction of the optimal solution for each class (630 classes). The image has been divided into two parts separated by a vertical dashed line. On the left, we have the classes without local optima and, on the right, we have the classes with one local optimum. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

the algorithm converge to the second best solution (which is not a local optimum). According to Corollary 2, the algorithm cannot converge to solutions which are not local optima. This anomalous behavior of the algorithm is due to the limited range of numbers that a computer can manage. Numerical checks reveal that, in certain generations of a reduced number of executions, the EDA assigns 0 to the probability of a solution whose probability is very close to 0 but not exactly 0. Therefore, the run degenerates and the result can be deceptive. These executions are isolated cases and are due to an intricate interaction between the initial point, the function and the probabilistic model. We do not try to correct these specific numerical errors in order to show the original results without additional manipulation of the simulations.

On the other hand, for the Univariate EDA (see Fig. 11), 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 to classes with one local optimum. It is the zone in which we can see the widest range of colors. When an arc is introduced in the probabilistic model used by the EDA (see Fig. 12), we can observe that the number of easy classes increases and that the hardest classes for the algorithm have a higher basin of attraction. In this specific scenario of three binary variables, we have the same number of classes for the Univariate EDA and the Arc EDA though, in general, they generate a different number of classes as can be verified by means of Eq. (18). When two arcs are added to the factorization creating a chain, the number of easy problems dramatically increases as can be seen in Fig. 13. In addition, the smallest basin size is clearly higher than in Figs. 11 and 12 and therefore, the hardest problems are not as difficult as in the previous algorithms.

Finally, regarding the V EDA, we can say that all the problems are easy for this algorithm. Nevertheless, we have different classes and hence, different behaviors. Although the algorithm always converges to the optimum, this convergence can occur at different speeds as we will see in the next section. The simulations of the V EDA suggest that this type of probabilistic model is able to guarantee the convergence to the optimum. To the best of our knowledge, there are no theoretical proofs regarding the convergence of EDAs that introduce a probabilistic model whose structural component is a V-structure.

Note that in general, and as expected, when the complexity of the factorization is increased, the algorithm has a greater overall ability to solve optimization problems.

8.2.2. 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 without local optima (in the case of the V EDA we consider all the classes). According to the basins of attraction, all these classes have the same complexity for the algorithm. However, we can observe in Fig. 14 different convergence behaviors. Starting from the uniform distribution, we show the curves that the probability of the optimum depicts throughout the generations for the Univariate EDA (Fig. 14(a)), Arc EDA (Fig. 14(a)), the Chain EDA (Fig. 14(b)) and the V EDA (Fig. 14(c)). We can see how, depending on the class, the same algorithm can have a slower or faster convergence. For instance, a slower convergence to the optimum can be interpreted as a consequence of facing harder problems. In addition, we can observe a narrower range of behaviors as the complexity of the model increases. Nevertheless, more complex EDAs do not necessarily imply a faster convergence.

9. A note on non-injective functions

Although the paper has been devoted to characterizing the behavior of estimation of distribution algorithms for injective functions, in this last section we would like to show how this characterization can be extended to non-injective functions. However, the complete characterization is beyond the scope of the paper, and therefore, we will introduce the basic framework with some general definitions and theorems, together with an illustrative example.

Given a non-injective function $f(\boldsymbol{x})$, we can naturally induce a partial ranking π of the solutions of the search space $\mathcal{X}$. This partial ranking π is ordered according to the function values $f(\mathcal{X})$ in such a way that:

i) $\pi(i) \succ \pi(j) \iff f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)$
ii) $\pi(i) \sim \pi(j) \iff f(\boldsymbol{x}_i) = f(\boldsymbol{x}_j)$

In the case of an injective function, the generated raking is not a partial ranking but a total ranking where the solution $\pi(1)$, namely $\boldsymbol{x}_1$, has the highest function value, $\pi(2) = \boldsymbol{x}_2$ has the second highest and so on, with the last solution $\pi(|\mathcal{X}|) = \boldsymbol{x}_{|\mathcal{X}|}$ being the one with the lowest function value:

$$f(\pi(1)) > f(\pi(2)) > \ldots > f(\pi(|\mathcal{X}|)).$$

Thus, in this case we interpret that $\pi(i) = \boldsymbol{x}$ returns the solution $\boldsymbol{x} \in \mathcal{X}$, which is at position i in the ranking.

In the case of non-injective functions, the generated ranking is not a total ranking and, because of that, there exist different representations π for the function, all of them representing the same partial ranking. The solution $\pi(1)$ corresponds to the solution $\boldsymbol{x}^*$ that solves Eq. (1). Independently of the specific function values $f(\mathcal{X})$, whenever they provide the same partial ranking of the solutions $\boldsymbol{x} \in \mathcal{X}$, the function is represented by the same partial ranking π. From now on, the objective function $f(\boldsymbol{x})$ and the corresponding ranking π can be used indistinctly as synonyms.

In Table 5, we provide a representation of a partial ranking π for the function $f(\boldsymbol{x})$. In the first column, the original function values are shown. In the second, we represent the solutions according to the function and, in the third column,

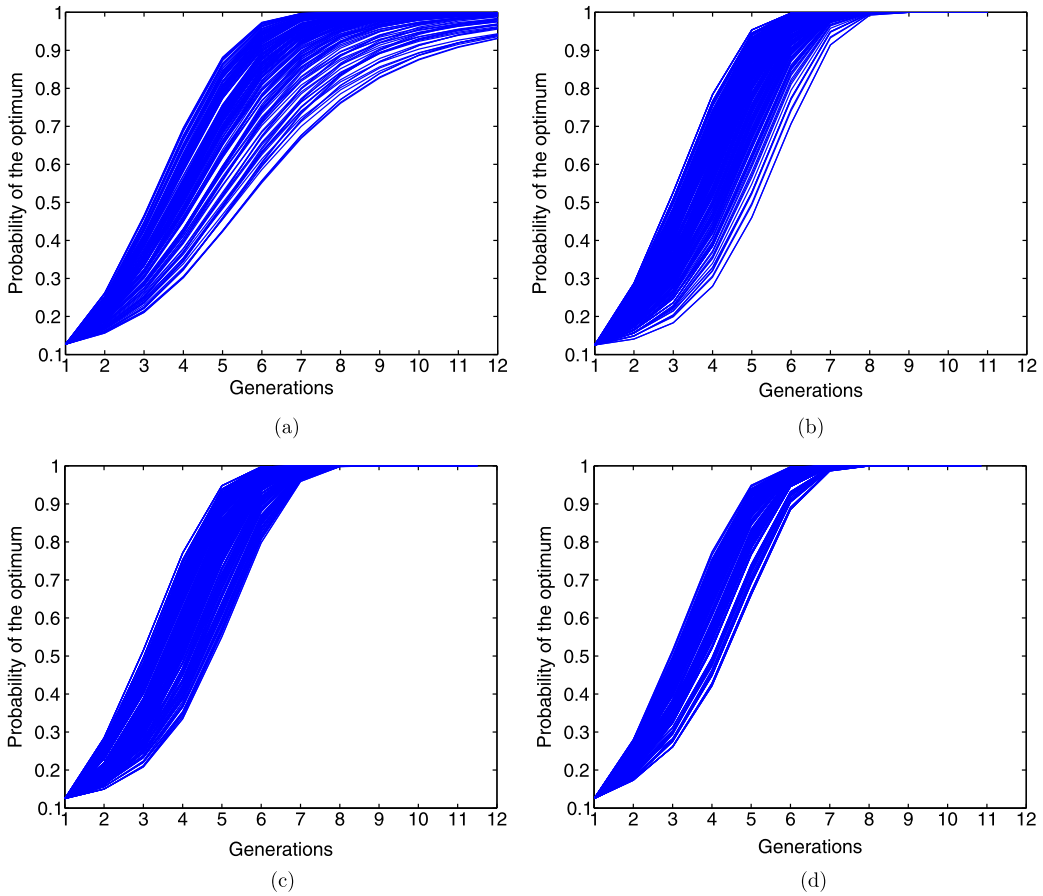


Fig. 14. Probability of the optimum along the generations. (a) Univariate EDA. (b) Arc EDA. (c) Chain EDA. (d) V EDA.

Table 5
Example of the representation of a partial ranking π for the function $f(x)$ for $n = 3$.

$f(\boldsymbol{x})$	π	i
100	(1, 1, 1)	1
100	(0, 1, 0)	2
100	(0, 0, 1)	3
100	(1, 0, 0)	4
100	(0, 1, 1)	5
100	(1, 0, 1)	6
68	(1, 1, 0)	7
23	(0, 0, 0)	8

we explicitly indicate the rank i of the solutions. As can be seen, the first six solutions can be ranked in any order (they all have the same function value), followed by the solutions in the seventh and eighth positions.

We can now provide a definition of equivalence for (non-injective and injective) functions, based on the definition provided for the case of injective functions (Definition 1). However, we need to define the completion of partial ranking previously.

Definition 2. Let π be a partial ranking, a total ranking $\overline{\pi}$ is a completion of π if there is no i and j such that $\pi(i) \succ \pi(j)$ and $\overline{\pi}(j) \succ \overline{\pi}(i)$.

The basic idea of a completion of a partial ranking is to assign an order to those solutions that are not ordered by keeping the order of the partial ranking. Now we can give the definition of equivalence in the general case.

Table 6
All the possible assignments for the last two positions of the ranking shown in Table 5. For the particular case of a univariate EDA, assignments are grouped in three equivalence classes red–bold, blue–italic, and green–underlined, with sizes 24, 24, and 8 respectively.

000	**000**	*000*	**000**	*000*	*000*	000
001	**010**	*011*	**100**	*101*	*110*	111
001	*001*	**001**	*001*	**001**	001	*001*
000	*010*	**011**	*100*	**101**	110	*111*
010	*010*	**010**	*010*	010	**010**	*010*
000	*001*	**011**	*100*	101	**110**	*111*
011	**011**	**011**	011	*011*	*011*	**011**
000	**001**	**010**	100	*101*	*110*	**111**
100	*100*	*100*	100	**100**	**100**	*100*
000	*001*	*010*	011	**101**	**110**	*111*
101	**101**	101	*101*	**101**	*101*	**101**
000	**001**	010	*011*	**100**	*110*	**111**
110	110	**110**	*110*	**110**	*110*	**110**
000	001	**010**	*011*	**100**	*101*	**111**
111	*111*	*111*	**111**	*111*	**111**	**111**
000	*001*	*010*	**011**	*100*	**101**	**110**

Definition 3. Let π_1 and π_2 be the (partial) rankings 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 π_1 and π_2 are equivalent under $\mathcal{A}$, and by extension f_1 and f_2, if for any completion $\overline{\pi_1}$ of π_1, there exists a completion $\overline{\pi_2}$ of π_2 such that $\overline{\pi_1}$ and $\overline{\pi_2}$ are equivalent under Definition 1.

Theorem 1 could be extended in the following way:

Theorem 3. *Let $\mathcal{A}$ be an EDA that implements $\mathcal{M}$ as $p^a(\boldsymbol{x}) = \prod_{i=1}^{n} p^s(x_i|\boldsymbol{pa}_i)$. Two functions π_1 and π_2 are equivalent under $\mathcal{A}$ if and only if for all completion $\overline{\pi_1}$ of π_1 and its corresponding completion $\overline{\pi_2}$ of π_2, the corresponding sets $G_{\overline{\pi_1}}$ and $G_{\overline{\pi_2}}$ are equal.*

This theorem is a generalization of Theorem 1, defined for injective functions. Instead of having two rankings, we depart from the more general case of two partial rankings, and each partial ranking will generate a set of completions (different representations). The proof of Theorem 3 is straightforward given the proof of Theorem 1.

Now that the corresponding definition and theorem have been introduced, a taxonomy of non-injective functions can be defined, calculating 1) the number of different functions (partial rankings) and, 2) for each function and probabilistic model, the number and size of the different equivalence classes. However, as the reader can imagine, non-injectivity makes the number of different functions significantly larger, and the identification and counting of equivalence classes becomes harder. Therefore, these aspects are postponed for future research.

Nevertheless, we end this section by illustrating the counting of the number of equivalent classes and their size for the (partial) ranking provided in Table 5. Taking into account that the first 6 solutions must have the same value, the total number of different rankings is given by the positions with strictly different rankings. In this case, only the last two solutions that correspond with the two lowest values of the function. Therefore, only 56 rankings (related to the combination of different solutions in this last two positions) can be defined. Table 6 shows all the possible combinations.

Once the total number of rankings has been calculated, the equivalence classes they are grouped in can be obtained. In this example, we use a univariate model. Then, according to Theorem 3, two functions π_1 and π_2 are equivalent under $\mathcal{A}$ (given model $\mathcal{M}$) if and only if for all completion $\overline{\pi_1}$ of π_1 and its corresponding completion $\overline{\pi_2}$ of π_2, the corresponding sets $G_{\overline{\pi_1}}$ and $G_{\overline{\pi_2}}$ are equal. Let us suppose that $\overline{\pi_1}$ refers to the completion (total ranking) shown in Table 5. Then,

$$G_{\overline{\pi_1}} = \left\{ \begin{pmatrix} 1 & 4 & 6 & 7 \\ 2 & 3 & 5 & 8 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 5 & 7 \\ 3 & 4 & 6 & 8 \end{pmatrix} \begin{pmatrix} 1 & 3 & 5 & 6 \\ 2 & 4 & 7 & 8 \end{pmatrix} \right\}.$$

As many different completions can be defined over π_1, for the sake of simplicity, let us use an $*$ for those value assignments that could appear in different positions. Then, G_{π_1} could be defined as:

$$G_{\pi_1} = \left\{ \begin{pmatrix} * & * & * & 7 \\ * & * & * & 8 \end{pmatrix}, \begin{pmatrix} * & * & * & 7 \\ * & * & * & 8 \end{pmatrix} \begin{pmatrix} * & * & * & * \\ * & * & 7 & 8 \end{pmatrix} \right\},$$

where only those assignments given to the last two positions of the ranking are really taken into account. Now, if we consider a ranking π_2, which ranks solutions $(0, 1, 0)$ and $(0, 0, 1)$ in the seventh and eighth positions respectively, we could check that matrix G_{π_2} is equal to G_{π_1}. Repeating this process for all the possible assignments, the total number of equivalence classes and their sizes is obtained. Table 6 shows all the equivalence classes and their respective sizes for the type of function and model used in the example. Particularly, there are three different classes (red–bold, blue–italic, and green–underlined), their respective sizes being 24, 24, and 8. As can be observed, another difference with respect to the injective case is that equivalence classes of the same size are not guaranteed.

10. Conclusions

The current paper, which continues the research line initiated in [23,38], provides significant advances in four main aspects regarding the taxonomy of problems and the behavior of EDAs: i) the equivalence condition is extended to EDAs whose factorization is given by any Bayesian network, ii) the description of the functions belonging to a class is developed for factorizations given by chordal Markov networks, iii) the relationship between neighborhood systems and this type of factorizations is established and iv) the numerical simulations show all possible behaviors that an EDA implementing a Bayesian network of three variables can exhibit.

In order to establish the equivalence condition for EDAs implementing a Bayesian network, we have provided a new matrix representation of the relationship between the function and the factorization. Taking into account this representation, we have proved the necessary and sufficient condition to determine the equivalence between functions and to partition the space of problems.

Two major contributions of this paper are the description of the equivalent functions and the relationship between equivalence classes and neighborhood systems. In order to describe the equivalent functions when the EDA implements a chordal Markov network, we have synthesized the basic transformations that can be made in a ranking π in order to generate new equivalent functions and count the members of a class. Note that the transformation used to generate equivalent functions under EDAs are, in essence, symmetries in the search space determined by a probabilistic graphical model. This point of view can be useful to study the relationship between optimization problems and probabilistic models from an algebraic perspective, independently of the specific optimization algorithm.

Although our description of the equivalent functions has focused on chordal graphs, we notice that these graphs comprise models that can be represented by Bayesian networks but also by Markov networks, another type of probabilistic graphical model applied in EDAs. A substantial amount of research in EDAs has been conducted on models with chordal structure, which provide an "exact" factorization of the distribution by means of a junction tree. Examples of the algorithms that use these models in EDAs are the factorized distribution algorithm [25], the extended compact genetic algorithm (ECGA), [34], and more recently the Affinity EDA [39]. Our work is relevant for the analysis of this type of EDAs.

Regarding the relationship between EDAs and neighborhood systems, we have presented a distance between solutions associated to the factorization implemented by the EDA. Given an EDA and the associated distance, we know that all the functions in a class have the same number of local optima and in the same ranking positions. Moreover, we know that the EDA cannot converge to solutions which are not local optima according to this distance. These facts reveal an intrinsic connection between EDAs and neighborhood systems defined on the search space. It suggests that the factorization used by the EDA can induce a structure in the fitness landscape that could be exploited by the algorithm during the search. The distance associated to the factorization could be used to implement local search operators and improve the performance of the EDA. It is also important to note that more complex factorizations imply a lower number of local optima in the function and therefore, a higher chance of converging to the global optimum. This phenomenon is reflected in the numerical analysis of the algorithm.

The experimental part presents a complete view of the different behaviors that an EDA implementing a Bayesian network can exhibit. We have observed that increasing the complexity of the model improves the overall ability of the algorithm to reach the optimum. In addition, we have seen that the speed of convergence could be a useful descriptor of the difficulty of the problems for the EDA and it should be taken into account for further research.

Finally, we discuss the connections that emerge between EDAs and other disciplines. On the one hand, some of these connections had already been previously established in the field of EDAs by different authors. Here, we can point out the use of dynamical systems to the study of EDAs and the close relationship between the behavior of the algorithm and neighborhood systems defined on the search space. On the other hand, this paper also reveals novel connections between EDAs and other mathematical fields. In particular, the equivalence classes can be seen as symmetries of the search space and therefore, they could be studied by means of group theory. From this perspective, we can consider a permutation $\sigma : \mathcal{X} \to \mathcal{X}$ which, applied to any function π, can generate an equivalent function π' as $\sigma(\pi(i)) = \pi'(i)$. Thus, the equivalent functions determined by a given factorization could be modeled as a subgroup of the symmetric group. In addition, the theory of graph automorphisms is useful to obtain a specific subset of equivalent functions associated to the structural component of the probabilistic model implemented by the EDA. Also, the neighborhood systems can be studied under the light of group theory and therefore, they could be connected to EDAs through this path. The further development of this theoretical framework could lay the foundations of a novel theory of EDAs. In particular, an important topic for further research is the in-depth analysis of non-injective functions. As shown before, the theoretical foundations to study non-injectivity are based on the theory presented in the current paper.

Acknowledgements

This work has been partially supported by IT-609-13 programs (Basque Government), and TIN2013-41272P (Spanish Ministry of Economy and Competitiveness MINECO).

References

[1] M. Zlochin, M. Birattari, N. Meuleau, M. Dorigo, Model-based search for combinatorial optimization: a critical survey, Ann. Oper. Res. 131 (2004) 373–395.
[2] J.A. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea (Eds.), Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms, Springer-Verlag, 2006.
[3] D.H. Wolpert, D. Rajnarayan, Using machine learning to improve stochastic optimization, in: Late-Breaking Developments in the Field of Artificial Intelligence. Papers Presented at the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI Press, Palo Alto, California, 2013.
[4] Z. Botev, D.P. Kroese, R.Y. Rubinstein, P. L'Ecuyer, The cross-entropy method for optimization, in: Machine Learning: Theory and Applications, in: Handbook of Statistics, vol. 31, 2013, pp. 35–59.
[5] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions I. Binary parameters, in: Parallel Problem Solving from Nature, PPSN IV, in: Lecture Notes in Computer Science, vol. 1141, Springer-Verlag, Berlin, 1996, pp. 178–187.
[6] P. Larrañaga, J.A. Lozano (Eds.), Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.
[7] M. Pelikan, Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms, Studies in Fuzziness and Soft Computing, Springer, 2005.
[8] P.A. Bosman, 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, ACM Press, 2010, pp. 351–358.
[9] M. Hauschild, M. Pelikan, K. Sastry, D.E. Goldberg, Using previous models to bias structural learning in the hierarchical BOA, Evol. Comput. 20 (2012) 135–160.
[10] P. Larrañaga, H. Karshenas, C. Bielza, R. Santana, A review on probabilistic graphical models in evolutionary computation, J. Heuristics 18 (2012) 795–819.
[11] R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J.L. Flores, J.A. Lozano, Y. Van de Peer, R. Blanco, V. Robles, C. Bielza, P. Larrañaga, A review of estimation of distribution algorithms in bioinformatics, BioData Min. 1 (2008) 1–12.
[12] R. Santana, P. Larrañaga, J.A. Lozano, Protein folding in simplified models with estimation of distribution algorithms, IEEE Trans. Evol. Comput. 12 (2008) 418–438.
[13] A. Brownlee, M. Pelikan, J. McCall, A. Petrovski, An application of a multivariate estimation of distribution algorithm to cancer chemotherapy, in: Proceedings of the 10th Genetic and Evolutionary Computation Conference, GECCO-2008, ACM, 2008, pp. 1033–1040.
[14] Q. Zhang, On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm, IEEE Trans. Evol. Comput. 8 (2004) 80–93.
[15] J.L. Shapiro, Drift and scaling in estimation of distribution algorithms, Evol. Comput. 13 (2005) 99–123.
[16] C. Echegoyen, Q. Zhang, A. Mendiburu, R. Santana, J.A. Lozano, On the limits of effectiveness in estimation of distribution algorithms, in: Proceedings of the 2011 Congress on Evolutionary Computation, CEC-2011, IEEE Press, New Orleans, USA, 2011, pp. 1573–1580.
[17] E.A. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Natural Computing Series, Springer, 2003.
[18] C. González, J.A. Lozano, P. Larrañaga, Mathematical modeling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions, Internat. J. Approx. Reason. 31 (2002) 313–340.
[19] Q. Zhang, H. Mühlenbein, On the convergence of a class of estimation of distribution algorithms, IEEE Trans. Evol. Comput. 8 (2004) 127–136.
[20] T. Chen, K. Tang, G. Chen, X. Yao, Analysis of computational time of simple estimation of distribution algorithms, IEEE Trans. Evol. Comput. 14 (2010) 1–22.
[21] M. Hauschild, M. Pelikan, K. Sastry, C. Lima, Analyzing probabilistic models in hierarchical BOA, IEEE Trans. Evol. Comput. 13 (2009) 1199–1217.
[22] C. Echegoyen, A. Mendiburu, R. Santana, J.A. Lozano, Towards understanding EDAs based on Bayesian networks through a quantitative analysis, IEEE Trans. Evol. Comput. 16 (2012) 173–189.
[23] C. Echegoyen, A. Mendiburu, R. Santana, J.A. Lozano, On the taxonomy of optimization problems under estimation of distribution algorithms, Evol. Comput. 21 (2013).
[24] E. Castillo, J.M. Gutierrez, A.S. Hadi, Expert Systems and Probabilistic Network Models, Springer-Verlag, 1997.
[25] H. Mühlenbein, T. Mahnig, A. Ochoa, Schemata, distributions and graphical models in evolutionary optimization, J. Heuristics 5 (1999) 213–247.
[26] S. Shakya, R. Santana (Eds.), Markov Networks in Evolutionary Computation, Springer, 2012.
[27] Y. Gao, J.C. Culberson, Space complexity of estimation of distribution algorithms, Evol. Comput. 13 (2005) 125–143.
[28] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison–Wesley, Reading, MA, 1989.
[29] R. Santana, P. Larrañaga, J.A. Lozano, Research topics on discrete estimation of distribution algorithms, Memetic Comput. 1 (2009) 35–54.
[30] M.D. Vose, The Simple Genetic Algorithm: Foundations and Theory, MIT Press, 1999.
[31] B. Doerr, C. Winzen, Ranking-based black-box complexity, Algorithmica 68 (3) (2014) 571–609, http://dx.doi.org/10.1007/s00453-012-9684-9.
[32] C. González, J.A. Lozano, P. Larrañaga, Analyzing the PBIL algorithm by means of discrete dynamical systems, Complex Systems 12 (2001) 465–479.
[33] H. Mühlenbein, The equation for response to selection and its use for prediction, Evol. Comput. 5 (1998) 303–346.
[34] G. Harik, Linkage learning via probabilistic modeling in the ECGA, IlliGAL report 99010, University of Illinois at Urbana–Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.
[35] S.L. Lauritzen, Graphical Models, Clarendon Press, Oxford, 1996.
[36] A. Ochoa, M.R. Soto, R. Santana, J. Madera, N. Jorge, The factorized distribution algorithm and the junction tree: a learning perspective, in: A. Ochoa, M.R. Soto, R. Santana (Eds.), Proceedings of the Second Symposium on Artificial Intelligence, CIMAF-99, Havana, Cuba, 1999, pp. 368–377.
[37] A.H. Wright, S. Pulavarty, Estimation of distribution algorithm based on linkage discovery and factorization, in: H.-G. Beyer, U.-M. O'Reilly (Eds.), Proceedings of Genetic and Evolutionary Computation Conference, GECCO-2005, ACM, Washington, D.C., USA, 2005, pp. 695–703.
[38] C. Echegoyen, Contributions to the analysis and understanding of estimation of distribution algorithms, Ph.D. thesis, University of the Basque Country, 2012.
[39] R. Santana, C. Bielza, J.A. Lozano, P. Larrañaga, Mining probabilistic models learned by EDAs in the optimization of multi-objective problems, in: Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference, GECCO-2009, ACM, New York, NY, USA, 2009, pp. 445–452.