Natural Computing (2024) 23:99–113
https://doi.org/10.1007/s11047-022-09913-2

A roadmap for solving optimization problems with estimation of distribution algorithms

Josu Ceberio[1] · Alexander Mendiburu[2] · Jose A. Lozano[1]

Accepted: 25 July 2022 / Published online: 6 September 2022

Abstract
In recent decades, Estimation of Distribution Algorithms (EDAs) have gained much popularity in the evolutionary computation community for solving optimization problems. Characterized by the use of probabilistic models to represent the solutions and the interactions between the variables of the problem, EDAs can be applied to either discrete, continuous or mixed domain problems. Due to this robustness, these algorithms have been used to solve a diverse set of real-world and academic optimization problems. However, a straightforward application is only limited to a few cases, and for the general case, an efficient application requires intuition from the problem as well as notable understanding in probabilistic modeling. In this paper, we provide a roadmap for solving optimization problems via EDAs. It is not the aim of the paper to provide a thorough review of EDAs, but to present a guide for those practitioners interested in using the potential of EDAs when solving optimization problems. In order to present a roadmap which is as useful as possible, we address the key aspects involved in the design and application of EDAs, in a sequence of stages: (1) the choice of the codification, (2) the choice of the probability model, (3) strategies to incorporate knowledge about the problem to the model, and (4) balancing the diversification-intensification behavior of the EDA. At each stage, first, the contents are presented together with common practices and advice to follow. Then, an illustration is given with an example which shows different alternatives. In addition to the roadmap, the paper presents current open challenges when developing EDAs, and revises paths for future research advances in the context of EDAs.

Keywords Estimation of distribution algorithm · Optimization · Problem · Probabilistic model · Domain

1 Introduction

Since the first works by Mühlenbein, Mahnig and Paaß (1998, 1999, 1996), Estimation of Distribution Algorithms (EDAs) have consolidated as a solid paradigm for optimization in the evolutionary computation field. Due to their potential, a vast number of EDAs works have approached optimization problems in very diverse topics such as protein structure prediction (Santana et al. 2008), chemotherapy treatment optimization for cancer (Brownlee et al. 2008), environmental water monitoring (Kollat and Reed 2008), nuclear reactor fuel management parameter optimization (Jiang et al. 2006), and more recently, web service composition (Wang et al. 2018), sensor location in chemical plants (Carnero et al. 2018) and calibration of

This work has been partially supported by the , ELKARTEK program (KK-2020/00049) and Research Groups 2022–2025 (IT1504-22) from the Basque Government, the PID2019-106453GA-I00 and PID2019-104933GB-10 research projects from the Spanish Ministry of Science and Innovation.

Extended author information available on the last page of the article

✉ Josu Ceberio
josu.ceberio@ehu.eus

Alexander Mendiburu
alexander.mendiburu@ehu.eus

Jose A. Lozano
ja.lozano@ehu.eus

[1] Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Manuel Lardizabal Pasealekua 1, 20018 Donostia-San Sebastian, Spain

[2] Department of Computer Architecture and Technology, University of the Basque Country UPV/EHU, Manuel Lardizabal Pasealekua 1, 20018 Donostia-San Sebastian, Spain

traffic models (Fard and Mohaymany 2019). In addition to the previous references on the solution of real-world problems, a large number of papers have been published regarding theoretical features of EDAs. To name a few, Echegoyen et al. (2012, 2013) analyzed the behavior of EDAs on different types of problems, Zhang and Muhlenbein (2004) investigated the global convergence of a class of EDAs, Shapiro (2005) addressed the effects of drift and scaling in EDAs, and Krejca and Witt (2018) proposed a theoretical explanation of their dynamics. Motivated by the large amount of works published on the topic, a number of journal papers and books have surveyed the proposed designs and identified research lines for future investigations. Some relevant works are, amongst others, the general review paper by Pelikan et al. (2002), a paper on challenges and open problems by Santana et al. (2007), on bioinformatics by Armañanzas et al. (2008) or on permutation problems by Ceberio et al. (2012), and the books edited by Larrañaga and Lozano (2002); Lozano et al. (2006) and Pelikan et al. (2006). In response to the great usage of EDAs, specific software such as MATEDA (Santana et al. 2010) (a Matlab toolbox for EDAs) and extensions to it by Irurozki et al. (2018) have also been proposed.

Estimation of Distribution Algorithms were proposed as an extension of Genetic Algorithms (GA) (Goldberg 1989). GAs apply crossover and mutation operators to the set of selected solutions in order to create new solutions. The principle is that the combination of good solutions may lead to create better solutions, while combinations with bad solutions will probably provide low quality solutions. EDAs work on the same basis, however, contrarily to GAs, they learn a joint probability distribution from the set of most promising solutions at each iteration. The aim is to assign higher probability to those solutions (or parts of them) that are more frequently observed in the population. Then, a new iteration of solutions is obtained by sampling the probability distribution learned. The algorithm stops iterating and returns the best solution found across the generations when a certain stopping criterion is met, such as a maximum execution budget or the same probabilistic model is estimated for a number of consecutive iterations (in terms of either structure or parameters). Figure 1 introduces a general pseudo-code of EDAs.

The use of probabilistic models with population-based algorithms has meant a qualitative change in the field of evolutionary algorithms. On the one hand, probabilistic models allow information about the interactions between the problem components to be captured and summarized. In addition, probability models permit information about the problem to be explicitly incorporated, allowing a more efficient search of optimal solutions. On the other hand, EDAs build a sequence of probability models (one per iteration) that describes the path followed by the algorithm to reach that solution. This path can reveal previously unknown information about the structure of the optimization problem, or can be used to develop strategies that consider information about the optimization process followed.

As seen, EDAs have interesting properties, however obtaining successful designs is not trivial, and there are many aspects that, in this context, require special attention. The probability model is the core of an EDA and it greatly determines the performance of it. Particularly, the ability of the model to identify, capture and propagate high quality solutions is the key point. As a result, when a practitioner has a problem to optimize and aims to apply an EDA, usually, the main point during the design has been correctly answering the following question: *which is the most suitable probability model for optimizing my problem?* This question cannot be answered unless we address at the same time other aspects that are also relevant. For instance, aspects related to the encoding of the solutions, or the complexity of the model (either in the number of parameters or in the degree of interactions it models) cannot be addressed independently, and requires a joint analysis to determine the suitability of a probability model to solve the problem at hand.

Moreover, there are more aspects that influence the overall performance of the EDA. For example, incorporating problem specific information in the probabilistic model can be very valuable in the optimization, or controlling the diversification-intensification behavior of the algorithm by tuning the parameters of the model may severely affect the performance of EDAs. Although these two points are relative to any evolutionary algorithm, in the case of EDAs, developing strategies to approach them may involve dealing with the probability model, and are, for that reason, included in this paper.

In this sense, the first aim of this paper is to present a roadmap for designing EDAs to solve optimization problems by addressing the points presented above. We provide a schema to guide practitioners throughout the design process of an EDA explaining the key aspects that compromise the performance of the algorithm. In order to illustrate these aspects, together with theoretical considerations, examples from different domains of applications are provided. The second aim of the manuscript is to identify the possible research trends in EDAs that enhance a better understanding of the dynamics of EDAs, and also, to more efficient algorithmic designs.

The rest of the paper is organized as follows. In Sect. 2 the roadmap for designing EDAs is presented as a sequence of four stages. Afterwards, in Sect. 3, existing open research lines for the future are discussed. Finally, the paper concludes in Sect. 4.

1. $l \leftarrow 0$
2. $D_0 \leftarrow$ Generate and evaluate the initial population of solutions of size M
3. **While** stopping criterion **not** met **do**
4. $\quad D_l^{Se} \leftarrow$ Select $N \leq M$ individuals from D_l
5. $\quad p(x \in S|D_l^{Se}) \leftarrow$ Build a probabilistic model [a] from the chosen solutions D_l^{Se}.
5. $\quad U_l \leftarrow$ Sample M individuals (the new population) from $p(x \in S|D_l^{Se})$ and evaluate.
6. $\quad D_{l+1} \leftarrow$ Build new generation population from D_l and U_l
7. $\quad l \leftarrow l + 1$
8. **End do**
9. **Return** best individual

[a] S denotes the space of samples onto which the probability distribution is defined.

Fig. 1 General outline of estimation of distribution algorithms (EDAs)

2 The roadmap for solving an optimization problem via EDAs

In this section, the main purpose of this paper, a roadmap for solving optimization problems by means of EDAs is presented. As introduced in previous sections, a large number of schemes have been developed using EDAs as a basis for solving different problems. Classically, when applying EDAs, the first step has been to identify the domain of the problem: combinatorial, continuous or mixed (not so usual). Then, the second step usually consists of deciding the type of the model to use considering various aspects such as parametric/non-parametric, complexity of the model (measured, for instance, in terms of the number of parameters) or the methods to learn and sample it. However, in addition to the previous points, there are many other decisions to make in between that have a meaningful impact on the performance of EDAs, and have not captured the required attention. In what follows, divided in four stages, we elaborate on the different aspects that are recommended to address when designing EDAs. Together with the previous notes, illustrative examples of the application of these algorithms on optimization problems are provided, which can help the reader compare the consequences of using the different alternatives in each case.

The contents introduced in the stages below depend, in many cases, on each other and it is difficult to make decisions separately. Bearing that in mind, we present the contents in an order which tries to be the most logical one for any practitioner that is not familiar with EDAs. Before starting, it is important for the reader to also note that, although some of the issues discussed in the roadmap are relevant for any evolutionary algorithm or metaheuristic, these have much more critical impact in the case of EDAs. Moreover, from the perspective of scientific research, the good practices recommended below are desirable to, first, understand the dynamics of EDAs, and, in general, to give steps towards future paradigmatic designs.

2.1 Stage 1: The choice of the codification

When approaching an optimization problem, the first step is to identify what a solution is and also which the set of all the feasible solutions is. A *solution* can be described in the form of a vector, matrix, graph or any other data structure. The set of all feasible solutions is usually called the search space. However for convenience, we will avoid using this term[1]. Conversely, we will call it *solution space*, denoted as Ω_S, and it is the set of solutions from which the optimization algorithm, in our case the EDA, ultimately should return one candidate.

Once the solution space has been identified, the next decision to make is to find a suitable representation to codify the solutions in Ω_S. Note that an encoding induces a space of elements that not necessarily matches Ω_S (see Example 1 for an illustration). Let us call this set the *codification space*, denoted as Ω, and *individual* to each element $y \in \Omega$. In general, this decision is made independently of the algorithm to be used on the problem. Nevertheless, from the viewpoint of EDAs, there are three points related with the encoding that are **relevant for the design of the probabilistic model**, and are desirable Ω to hold:

1. For any solution $x \in \Omega_S$, there should exist an individual $y \in \Omega$ that represents x.
2. For any individual $y \in \Omega$, there should exist a corresponding solution $x \in \Omega_S$ of the problem. This fact implies that any $y \in \Omega$ describes a feasible solution for the problem.
3. There should exist a bijection between Ω_S and Ω. Even when the two points above are held, it is possible to have redundancies between the solution space, and codification space can make accurate probability model learning impossible as several individuals of Ω

[1] The term *search space* has been used in a great number of optimization papers with very different meanings. Since different spaces are considered in this paper, in order to avoid confusions produced by already-held beliefs, we decided to avoid that term.

represent the same solution in Ω_S. In order to avoid this happening, any solution in Ω_S should only be described by a unique individual in Ω, and any individual in Ω should only describe one solution in Ω_S.

In essence, the encoding should permit to discriminate between good and bad quality solutions. This way the algorithm is able to, given an encoding for the solutions, identify the features that determine their quality. It is harmful to provide an encoding that generates large redundancies since it enables the same solution to be represented with different individuals (different description). This confuses the EDA as it is not able to determine the features of individuals that characterize their quality. Unless the probability model is very flexible, it is a difficult task which the EDA is unlikely to address successfully.

In addition to the solution and codification spaces, there is a third space to incorporate to the discussion that is specific for the case of EDAs. As described in the introduction, EDAs search solutions by learning at every iteration a probability model, and then, by sampling potential solutions from the probability distribution induced by the model. In other words, EDAs return *samples* from a space onto which the probability distribution is defined. From probability theory, this space is known as the *sample space*, and we denote it S.

Ideally, it is desirable to hold $\Omega = S$, that is, the probability distribution is defined on a sample space that equals the codification space. This way, any sample obtained corresponds to an individual of Ω, and if defined appropriately, to a solution of the problem, $\Omega_S = \Omega = S$. Unfortunately, in many occasions $\Omega \neq S$, and thus, a frequent alternative is to adapt the sampling procedure in order to generate samples that are individuals in Ω. This option, despite being practical, goes against the philosophy of EDAs as the samples obtained at each generation do not follow the estimated distribution, but another underlying and unknown one. As we will explain in the second stage, our advice is to hold $\Omega_S = \Omega = S$, and so, encourage the search for codifications that hold $\Omega_S = \Omega$, and for probability models that have $\Omega = S$ as their sample space. In what follows, Example 1 illustrates the three spaces above and shows that, holding $\Omega = S$, sometimes is not easy.

Example 1 **Solution, codification and sample spaces in the Travelling Salesman Problem**. Let us consider, with illustrative purposes, one of the most recurrent combinatorial optimization problems in the literature: the Travelling Salesman Problem (TSP) (Goldberg and Lingle 1985). The TSP is the problem of finding the route to go over n different cities visiting each city only once and returning to the city of departure. The goal is to find the route that minimizes the time, cost or any other similar criteria associated to it. By considering the most simple case of the problem, the *symmetric* TSP,[2] the solution space Ω_S is composed of the set of all possible routes being $|\Omega_S| = \frac{(n-1)!}{2}$. We can think of this set as the different Hamiltonian cycles on a complete graph where cities are vertices and edges each have a weight associated that describes the cost of going from one city to another.

According to the literature on the TSP, depending on the approach used to solve it, different encodings have been considered to codify the solutions. A classical encoding is that which codifies routes as permutations of n cities.[3] Under this encoding, given any route σ (permutation of cities), $\sigma(i) = j$ represents that the j^{th} city is visited between the visits to the $\sigma(i-1)^{th}$ and $\sigma(i+1)^{th}$ cities. For instance, in a TSP of $n = 4$ cities, the permutation $\sigma = 3241$ describes the route that from city 3, then goes through 2, 4 and 1, and returns back to 3. Assuming the permutation encoding, the codification space Ω consists of the group of all permutations of size n:

$$\Omega = \mathbb{S}_n$$

and it is composed of $n!$ permutations. Under this encoding, any $\sigma \in \Omega$ corresponds to a unique route in Ω_S, unfortunately, for any route $x \in \Omega_S$ there are $2n$ different permutations. Following the example above, the permutation $\sigma' = 1324$ describes the same route as σ, as the starting city can be any of the n possible ones. In other words, this encoding is redundant. Note that under this encoding, the third property of the desired and enumerated above does not hold.

Let us accept the previous codification, then a suitable choice of the probability model can be that proposed by the Univariate Marginal Distribution Algorithm (UMDA) (Mühlenbein and Paaß 1996) which assumes that the variables of the problem are independent. In the case of the TSP, an application of this EDA calculates a matrix of probabilities that describes the probability of any integer (city) in $\{1, 2, \ldots, n\}$ appearing at each position. Then, the sample space onto which the probability distribution is defined is

$$S = \{1, \ldots, n\}^n.$$

The computation of the probability matrix is carried out by computing the number of times each city appears at each position across the selected solutions in the population divided by the number of individuals. Then, the probability of any sample $x \in S$ is calculated as

[2] In this TSP variant, the cost involved of travelling between cities i and j is equal in either direction.

[3] A *permutation* is understood as a bijection σ of the set of natural numbers $\{1, \ldots, n\}$ onto itself.

$$P(x) = \prod_{i=1}^{n} p_i(x_i) \tag{1}$$

where $p_i(x_i)$ denotes the probability of city x_i appearing at position i. Below, an example of a probability matrix learned given a set of five permutations is shown:

			Positions			
3241			1	2	3	4
1234		1	0.6	0.0	0.2	0.2
4213	$\Longrightarrow$ City	2	0.0	0.8	0.2	0.0
1324		3	0.2	0.2	0.2	0.4
1243		4	0.2	0.0	0.4	0.4

Given the matrix of probabilities in the example, two considerations are noteworthy: (1) permutations σ and σ' describe the same route, however, different probabilities are assigned by the model, which are 0.0128 and 0.0096 (computed by Eq. 1), respectively,[4] and (2) as $S \not\subset \Omega$, any sample in S, is not necessarily in Ω.

The first consideration has to do with the redundancy generated by the encoding. Since a route is a cyclic structure, this redundancy cannot really be solved while restrictions regarding the starting city and the direction of the route are not added to the individuals. Regarding the second consideration, although UMDA is easy to implement and apply, in the case of TSP, it does not match with the philosophy of EDAs that we have already discussed. The UMDA has been defined over the combinatorial space $\{1, \ldots, n\}^n$, but not over permutations and, thus, $\Omega_S \neq \Omega$ under the considered encoding. For instance, based on the matrix in the example, let us consider the solutions with maximum probability, 1243 and 1244, according to the definition of Ω, only the first of both is a feasible route, however, both are in S.

In this sense, an alternative is to use a probability model that induces a distribution on Ω. Since the codification space Ω in this example stands for the *symmetric group* $\mathbb{S}_n$, then, an interesting option is to consider probability models for permutations in the framework of EDAs such as those based on either distances Fligner and Verducci (1986); Irurozki (2014) or based on order statistics by Marden (1995).

2.2 Stage 2: The choice of the probability model

One of the strengths of EDAs is the possibility to describe the optimization process as a sequence of probability models generated across the iterations. To that end, it is essential to use probability models that define distributions on sample spaces S that match the proposed codification space Ω. From a philosophical point of view, this is the natural line to follow in EDAs, however, it requires from the practitioner a deep knowledge and understanding of probability models (more than in other Evolutionary Computation researches). In this sense, one of the options consists of searching for a probability model that has already been reported in the probability and statistics theory, and whose sample space is Ω. In general, when the codification space Ω corresponds to an existing algebraic group/domain, then it is relatively straightforward to find a probability distribution defined on it. However, when Ω is a subset of a known domain/group (due to some restrictions), then this option is no longer valid. Thus, the last alternative implies designing specific probability models that hold $S = \Omega$.

In any of the cases above, there are a number of aspects to consider regarding the probability model to choose/implement. The complexity and flexibility of the model, the feasibility of the learning and sampling methods, or its interpretability are some of the relevant issues to pay attention to. It is worth noticing that, when solving a particular problem, it is difficult to accomplish with all the issues we enumerated above and elaborate in what follows. Nonetheless, we find it important to highlight those that have a meaningful impact on the algorithm, and should be taken into account when looking for a balanced design.

2.2.1 Defining the probability model

When defining a probability model in EDAs, it is important to focus on the ability of the model to identify the features that characterize high quality solutions, and then, on propagating them by assigning high probability. In the literature there is a wide variety of aspects to focus on when defining probability models. In what follows, we will focus on the following three: (1) the interactions among the variables to capture by the probability model, (2) the number of components of the model and (3) the number of parameters involved in it.

Since the first work on EDAs decades ago, **modeling interactions among the variables** (or subsets of variables) that define the individuals has been a key issue. In this sense, a recurrent option has been to implement probabilistic graphical models that, given the set of individuals, learn a structure of the dependencies between variables (assuming a certain degree of dependence, i.e., univariate, bivariate or multivariate) in the model, and calculate the associated conditional probabilities. This idea can be naturally implemented in either directed or undirected probabilistic graphical models such as Bayesian Networks (Etxeberria and Larrañaga 1999), Markov Networks

[4] We do not make any consideration regarding the sampling mechanism, and focus exclusively on the fact that a solution represented by two different individuals is assigned with different probabilities

(Shakya and Santana 2012), or Gaussian Networks (Lozano and Mendiburu 2002). An alternative and more recent way to deal with the interactions among the variables has been the copula approach, in which the statistical properties of the variables are modelled first, individually, and then, the dependencies are modelled using a copula (Soto et al. 2015). In some other cases, interactions are not captured by the learned structure but they are specified by means of an a priori defined factorization set by the practitioner (Mühlenbein and Mahning 1999; Santana et al. 2007) or using perturbation methods (Wright and Pulavarty 2005), usually based on some information obtained from the problem. Without being limited to the previous references, there are models that do not capture the interactions explicitly, but consider them implicitly in their definition. Examples of this are order statistic probability models (Thurstone 1927), distance-based exponential models (Critchlow and Fligner 1991) or other grid/lattice based models (Iyer 1950; Ceberio et al. 2017).

Despite the large amount of works in EDAs related to modeling interactions, it is not clear enough that the real effect of modeling different types of interactions among the variables in the performance of EDAs. When no information about the problem is at hand, i.e., black-box problems, then a large amount of papers have shown that modeling interactions can give successful results. However, let us consider an extreme example in which the codification space is $\{0,1\}^n$ and has the following objective function:

$$f(x) = \prod_{i=1}^{n} x(i) + \sum_{i=1}^{n} x(i). \tag{2}$$

In that problem, the n variables are related to each other due to the product term, however, optimizing f entirely relies on maximizing the additive term. modeling interactions in this case is not relevant at all. In view of that, it becomes relevant to study the objective function, and the codification space proposed, to identify the interactions whose information is relevant for optimizing.

Often the set of individuals from which to calculate the probability distribution is very heterogeneous. This makes it difficult to accurately account for any specific individual in the probability distribution, especially when parametric models are used. Let us consider a sample of heterogeneous individuals is selected, and the aim is to estimate the parameters of a unimodal probability model. In these cases, parametric models may fail due to lack of flexibility. At this point, a number of papers in the literature have proposed using **mixtures of probability models** in EDAs (Murphy and Martin Jan. 2003; Peña et al. 2005; Bosman and Thierens 2002), since they allow a heterogeneous population to be modeled by modeling it as a collection of homogeneous sub-populations. It is also possible to take that idea to the limit and implement **kernel models** where, around each individual in the set, one probability model is defined (Gallagher 2000; Lebanon and Mao 2008; Arza et al. 2020). Then, the overall probability model is a weighted sum of the models considered (one for each sub-population or individual). It is important to note that either mixture or kernel models are able to provide higher accuracy estimations than usual parametric models. Conversely, they may also provoke the overfitting effect which results in a lack of exploration ability of the EDA. For that reason, it is important to find a balance in the exploration-exploitation trade-off of the EDA when implementing these models successfully (this topic is revised in detail in Stage 4).

A final remark regarding the definition of the model has to do with the **number of parameters** involved in it. In general, probability models with a large number of parameters define probability distributions that accurately represent the given data. However, at the same time, the complexity of learning those parameters requires a greater effort, usually meaning a high number of data required to carry out the estimations. Leaving aside questions related to the benefits of using accurate models to the performance of EDAs (we will discuss this in latter sections in this stage), ans strictly focusing on the definition of the probability model, the recommendation is to find an accurate model defined with a low number of parameters.

2.2.2 Feasibility of learning and sampling methods

The community of statistics has posed a wide variety of probability models that can be included in EDAs. However, most of them do not have closed formulas for calculating the parameters or there only exist complex algorithmic approaches to sample from them. For instance, given a set of individuals, calculating the optimal structure and parameters of the Bayesian Network learned by EBNA (Etxeberria and Larrañaga 1999) or that of the Gaussian Network by EGNA (Lozano and Mendiburu 2002), is NP-hard, and thus, these are calculated by means of approximation methods. Similarly, when mixtures of models are implemented, it is mandatory to group individuals in homogeneous sub-populations. In these cases, clustering methods such as the Expectation-Maximization algorithm are applied (Bosman 2000), whose time-complexity is high. Similarly, in Ceberio et al. (2013), the Newton-Raphson algorithm is used to estimate the parameters of the probability model. In parallel to the learning methods, many sampling methods have also been designed following a similar trend. For instance, in EDAs that use Markov Networks (Shakya and McCall 2007; Shakya and Santana 2012; Alden and Miikkulainen 2016), Gibbs samplers are used for obtaining solutions from the model. Alternatively,

in Mendiburu et al. (2012); Höns (2012); Lima et al. (2009) propagation and message passing methods with the same purpose are considered.

Using approximation iterative methods such as those above results in excessive computation costs as they are run every iteration of the EDA. Not only that but, frequently these models do not change from iteration to iteration since the set of selected solutions does not change (especially when the algorithm has converged to a population of individuals). In the context of EDAs, using numerical methods introduces large overheads that in many cases are not desirable, especially when the stopping criteria are defined under the time cost. Another drawback that numerical methods entail has to do with stopping them. Usually, the estimations automatically return the result when the approximation difference between iterations is below a threshold α. However, based on which criteria should we choose α? Obviously, the lower the α, the more accurate estimations we obtain, nonetheless the implications of this decision to the performance of EDAs is not clear.

In addition to the previous drawbacks, it is common to observe numerical errors in the application of approximation methods that need to be handled at implementation time (i.e., divisions with zero). Note that the application of numerical methods are, in most of the cases, subject to a number of restrictions/assumptions related to the context of application (that rarely are evaluated) (Lange 2010).

Taking the previous comments into account, we recommend avoiding when possible the use of numerical approximation methods when developing EDAs, or analyzing their feasibility and applicability, if they are indispensable, by studying their convergence and sensitivity to numerical errors.

2.2.3 Interpretability

As previously discussed, one of the capabilities of EDAs is the ability to capture and summarize, at each iteration, the good features of solutions in the form of a probability distribution. As a result of the application of an EDA to a problem, a sequence of probability models is obtained, one model per each iteration of the algorithm. This sequence can be seen as the path followed by the algorithm to reach the final solution.

In optimization, the aim is to obtain the best quality solution possible, and thus, the practitioners are not usually interested in the path followed by the algorithm. However, frequently, when the knowledge about the problem is limited, then analyzing the path followed can be very valuable as it is possible to identify interactions between variables of the problem that good solutions contain. It is the knowledge obtained from the sequence of probability models that Santana et al. (2008) used to simulate the optimal folding of the proteins. Let us think, for example, of an EDA that estimates a Bayesian Network every iteration. Then, the edges that survive across the optimization identify interactions between the variables of the problem that characterize good solutions.

In order to extract information about the problem, and carry out processes as those mentioned, it is necessary to implement probability models that can be either understandable or *interpretable*. A counterexample to Bayesian Networks can be Neural Networks (NN). Despite their current success, the interpretability of these models is still limited, as well as our ability to justify when a NN-based approach either succeeds or fails.

2.2.4 Adequacy

Does addressing correctly the issues above guarantee an effective probability model for EDAs? Not really. When we are given a set of solutions from which to estimate a probability distribution, our aim is the probability model to be able to identify the aspects that determine the quality of a solution (Bosman and Grahl 2008). Even in the case that we are able to estimate the optimal probability distribution that best fits the set of solutions at hand, there is no guarantee that this will take the EDA to the optimum. Although preliminarily, the work by Echegoyen et al. (2007) on the exact learning of Bayesian Networks in EDAs revealed that more accurate estimations of parameters and structure of the graphical model leads the EDA to better preserve the features that characterize good solutions.

Previous notes are relevant to design and implement effective probability models within EDAs. However, it is important for the model to assign a higher probability to good solutions, and a lower probability to bad solutions. In this sense, some correlation between the objective function values and the probability distribution must exist to obtain a successful EDA. An inspiring probability distribution is the Boltzmann probability model

$$P(x) = \frac{e^{-\beta f(x)}}{\psi(\beta)} \tag{3}$$

that assigns probabilities as an exponential function in the objective value of the solutions (Kirkpatrick and Gelatt 1983), β describes the sharpness of the distribution, and $\psi(\beta)$ denotes the normalization function.

Example 2 **Choosing a probability model for the Graph Partitioning Problem**. In this example, we consider the 0–1 balanced Graph Partitioning Problem (GPP) (Mezuman and Weiss 2012). Under this problem, given a weighted graph of n vertices, the aim is to find two partitions of an equal number of vertices while the sum of the

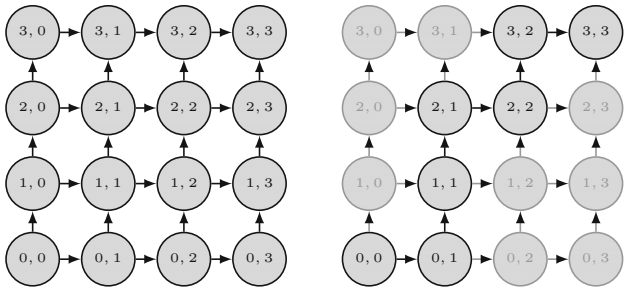
weights of the edges between the partitions, also known as *cutsize*, is minimized. Thus, solutions can be codified as binary vectors of an equal number of zeros and ones. Formally, $\Omega = \{x \in \{0,1\}^n | \sum_{i=1}^{n} x_i = n/2\}$. Given a GPP of $n = 6$ vertices, the individual $x = (0,1,1,0,0,1)$ represents a partition of the graph in which nodes 0, 3 and 4 are in one group, and 1,2 and 5 are in the other. The proposed codification is redundant, since, by negating all the positions in x, that is, $x' = (1,0,0,1,1,0)$, the same partition of vertices is described.

It becomes obvious that we will not easily find in the literature of probability and statistics a probability model whose sample space is the described Ω. Thus, below two different alternatives of probability models designed for solving the problem at hand (Ceberio et al. 2017, 2018), are described below. A generic option consists of implementing a distance-based exponential probability model whose sample space is Ω. Under this model, the probability value of every individual in the domain, $x \in \Omega$, is calculated as

$$P(x) \propto \exp(-\theta d(x, x_0))$$

where θ denotes the concentration of the distribution, $d(x, x_0)$ describes the distance from x to a reference individual x_0. The probability of any individual $x \in \Omega$ decreases exponentially with its distance from x_0. In order to implement this model, it is necessary to define a distance-metric whose domain is Ω. For this particular case, a metric based on the Hamming distance, which avoids redundancy of the sample space, was defined in Ceberio et al. (2018).

An alternative approach reported by the same authors proposes an EDA that implements a probability model defined within a square-lattice (see figures below) whose sample space is exactly Ω (Ceberio et al. 2017). Particularly, the lattice is described with $(n/2+1)^2$ vertices (in the figure below, $n = 6$), and individuals in Ω are modeled as paths from vertex (0, 0) to (n/2, n/2) in the lattice. Any movement in the path is given on the positive direction on either axis. As a result, any path that reaches point (h, k) is characterized by having, at stage $h + k$, exactly, h horizontal steps (number of zeros), and k vertical steps (number of ones). The individuals in Ω and can be rewritten as a path $x = \{(h^1, k^1), \ldots, (h^n, k^n)\}$ on the lattice where each term (h^i, k^i) denotes the point visited at stage i of the path. For instance, the lattice on the right describes the individual $x = (1,0,0,1,0,1)$ that can be rewritten as $x = \{(0,0),(0,1),(1,1),(2,1),(2,2),(3,2),(3,3)\}$.



A lattice of n = 6, and an example of a path that describes the individual x=(1,0,0,1,0,1)

Then, the probability distribution is codified by assigning to each point in the lattice a probability of giving a horizontal step, $P_{(h,k),(h+1,k)}$, or a vertical step $P_{(h,k),(h,k+1)}$. The overall probability of an individual x is calculated as the product of the probabilities of the movements given on the lattice path (see Eq. 4).

$$P(\pi) = \prod_{i=1}^{n} P_{(h^i,k^i),(h^i+1,k^i)}^{(h^{i+1}-h^i)} P_{(h^i,k^i),(h^i,k^i+1)}^{(k^{i+1}-k^i)} \quad (4)$$

Both models define a probability distribution over Ω, however they have notable differences with regard to the aspects elaborated in this stage. In Table 1, a summarized comparison is presented. *Which of the proposed approaches is better?* The practitioner should answer this question considering the context of application of the EDA.

Table 1 Comparison of the proposed models for the GPP according to the features revised in Stage 2

	Distance-based exponential model	Square-lattice model
Interactions	Probabilities are assigned to solutions in terms of their distance to a centroid x_0, so interactions are modelled implicitly.	The interactions between variables are modelled implicitly as solutions are described as paths in the lattice
Flexibility	The model is unimodal, so, low.	Not clear
Parameters	Two parameters (x_0 and θ)	$n^2/2$ parameters.
Feasibility	Numerical methods are required when learning MLE of parameters. Sampling is carried out with low time-complexity.	Low time-complexity methods are used when learning and sampling
Interpretability	Good, as the model has a centroid solution and a variance parameter.	Difficult interpretability
Adequacy	It depends on the correlation between the objective function and the distance-metric employed.	It is not clear how the paths are translated to the quality of solutions

2.3 Stage 3: Incorporating problem specific knowledge

When solving optimization problems, often we do not have information about its structure, thus, the EDA starts by initializing a population of candidate solutions randomly. Nevertheless, in a few cases, we do have information that can be helpful for the algorithm to find promising regions of solutions. One of the benefits of EDAs is that they allow the practitioner to incorporate information about the problem explicitly in the search. The most common way to do so consists of implementing a seeding method to generate a high quality initial population of solutions. This option biases the search favouring areas similar to the provided solutions, however, it is not exclusive to EDAs, and the initial population can be expected to be severely limited. An example in the framework of EDAs is given in Santana et al. (2010).

As EDAs estimate a probability distribution on the sample space, a second option is to bias or restrict the model building procedure (Hauschild et al. 2011). In this line, a number of papers have been published in the literature since 2000. Among which, Schwarz and Ocenasek (2000; Mühlenbein and Mahnig (2002) Baluja (2006) Santana et al. (2010) approached combinatorial problems whose definition is given on the basis of a graph structure (i.e., graph coloring problem). The authors realized that, when using EDAs based on graphical models (i.e., Bayesian networks...), it is possible to incorporate information obtained from a graph description of the problem directly to the model. Particularly, the prior knowledge provided a speed-up and better performance of the EDA. However, as Hauschild et al. (2011) noted, it may be difficult to process prior information about the problem structure to define adequate restrictions of model structure. Even if prior information about the problem structure is relatively straightforward to obtain and process, it may not be clear how to use this information to provide adequate model restrictions. In that context, Hauschild et al. (2011) propose some improvements to automatically biasing the model building in hBOA. In the same sense, Baluja (2006) argues that, although Bayesian Network or Markov Networks are the most representative probability models to be used when approaching problem with a graph structure, these models require more samples in order to build the dependency networks precisely.

Example 3 **Incorporating problem specific knowledge in the EDA**. As stated by Baluja (2006), learning probability models that are able to accurately model a high degree interdependencies between the variables that compose the problem has limitations. Despite their potential, they also require large amounts of data to accurately model the solution space, which is not usually available when running EDAs.

As a solution to that problem, Baluja proposed using information about the problem in order to direct the creation of the probabilistic network. The author showed that the interactions between the variables can be better ascertained from the data if prior knowledge on problem is available, and the modelled dependencies are reflective of real dependencies. With illustrative purposes, in Baluja (2006) the graph coloring problem was used as a case of study which is defined under a graph structure description.

The idea proposed constraints, according to the graph definition of the problem, the network structures that the EDA creates during the learning. Let us consider for illustration two nodes X and Y of a graph coloring problem instance. If X and Y are connected, then the algorithm will favour such a dependency by inserting an arc in the probability network regarding the two variables. Contrarily, if X and Y are not connected, then during the estimation of the probability network, any arc between them in the structure will be penalized.

The procedure to constrain the dependencies in the graph model can be carried out in different ways. One proposed by the author is to impose a prior probability over network structures in which the prior likelihood of a network decreases exponentially with the number of arcs in the network that do not correspond to edges in the problem structure. This is easily implemented by subtracting a penalty term to the mutual information computed between two variables when no edge exists between the two nodes in the graph description of the problem.

This work restricted the study to EDAs that learn a dependency tree that only models pair-wise interactions between the variables, however it is general to any EDA that learns network structure models De Bonet et al. (1996); Etxeberria and Larrañaga (1999); Pelikan et al. (2002). It is worth noting that the higher the connectivity of the problem instance, the higher the benefit of this strategy is supposed to be. Nevertheless, this approach to incorporate problem specific knowledge is limited to probability models that have explicit probability networks, and to problems that are defined by means of graph structures.

2.4 Stage 4: Setting up the algorithm: balancing the diversification-intensification trade-off

When running EDAs, we usually observe that in the initial iterations of the optimization, the trend is to see large improvements of the objective function values in the population of solutions. As the optimization progresses, the improvements become less frequent, until the algorithm converges to a solution (or set of solutions), and improvements are not likely anymore. This description of

the optimization process is recurrent, and can be observed for any metaheuristic algorithm.

Nevertheless, generally speaking, when designing an EDA, the aim is to diversify (or explore) in the initial steps of the optimization, in order to search for high quality areas of the solutions space. Then, once those areas are detected, in the following steps, the idea is to conduct the search towards those areas without ignoring other potential areas that can appear. Finally, the optimization focuses on specific high quality areas and intensifies (or exploits) the search.

Experience has shown that, detecting in which of the phases above the EDA is, at each iteration of the optimization, is nearly impossible. Moreover, when the algorithm has converged, there is not explicit information to decide whether the search was efficient, or it converged prematurely. However, balancing the diversification-intensification behavior of the algorithm is critical. In fact, a successful balancing can lead to great results, while failing in the same task can produce really bad performance.

The most recurrent problem regarding the trade-off is the lack of variance in the probability model as the optimization progresses. Usually, when the probability model employed within the EDA is efficient capturing and propagating the characteristics that define high quality solutions, these are assigned high probability in the probability distribution, and therefore, they are more likely to be sampled and survive in the next generation. This leads to concentrating the probability on that specific area of the search space by reducing the variance of the probability model. Unless that variance is lower-bounded, the next iteration is expected to have even lower variance, until all the probability is concentrated in the mode solution. This dynamic tends usually to what is known as *premature convergence*, and such lack of variance is one of the most typical problems in classical EDAs.

Conversely, when the initial population of solutions is generated uniformly at random, the diversity of the solutions may provoke the EDA to estimate probability distributions with large variance. This is frequent when unimodal shape models are used. Sampling models with such features is similar to sampling solutions uniformly at random, and therefore, the algorithm is not able to sample high quality solutions (as the distribution is nearly uniform).

What can we do to correctly balance the trade-off? There are multiple options, however, all of them try to guarantee an exploration behaviour of the EDA in the initial steps of the optimization, and as the optimization progresses, balance the EDA to an intensification behavior.

The most frequently observed strategy tries to tune the variance of the probability model directly. To that end, the strategies in this line use a parameter α that indicated the level of diversification/intensification the algorithm should have at that step (Ayodele et al. 2017). Alternatively, other works have considered using mixtures of probability models to balance the trade-off by adjusting the relevance of each of the components in the probability model based on the remaining budget for optimization (time, generations, evaluations...) (Alza et al. 2018). In Example 4, works from both domains are described.

Example 4 **Two strategies for balancing the diversification-intensification trade-off**. Ayodele et al. (2017) proposed a novel EDA for dealing with permutation-based problems: the Random Key Estimation of Distribution Algorithm (RK-EDA). This algorithm proposes using the Random Key (RK) representation to model permutation-coded solutions. Instead of using vectors of integer values or permutations, under this representation, a continuous encoding is employed which enables the use of probability models defined on the space of continuous vectors $\mathbb{R}^n$.

At each iteration of the EDA, a Gaussian distribution for each position of the permutations is considered. The best t_s solutions of the population are selected to generate a population S. Then, an array $\mu_{S_1}, \ldots, \mu_{S_n}$ that saves the mean of all RKs at indices $\{1, \ldots, n\}$ in the selected population S is computed. Note that μ_{S_n} refers to the mean of all RKs in the n^{th} index of each solution of S.

However, the variance is not estimated from the set of selected solutions (like mean μ_{S_n}), but it is initialized at the beginning of the optimization and a *cooling rate* c is computed to calculate the variance at each iteration v_g. The parameter c is calculated as

$$c = 1 - \frac{g}{MaxGen}$$

where g denotes the iteration and *MaxGen* stands for the maximum available generations. Multiplying c with the initial variance to form v_g makes it possible to achieve higher exploration at the start of the algorithm and more exploitation as g increases.

For the same problem domain, permutation-problems, in Alza et al. (2018) an EDA for solving the Linear Ordering Problem (LOP) is proposed. Particularly, the authors proposed using the Plackett-Luce probability model for modelling the selected solutions at each iteration of the algorithm. Under this model, given a vector of weights $\mathbf{w} = \{w_1, \ldots, w_n\}$ where w_i denotes the probability of item i appearing at the first position of the permutation, the probability of any permutation σ is calculated as

$$P(\sigma|\mathbf{w}) = \prod_{i=1}^{n-1} \frac{w_{\sigma(i)}}{\sum_{j=i}^{n} w_{\sigma(j)}}$$

Despite the promising results, the authors observed that the Plackett-Luce EDA suffers premature convergence

because of a drastic drop of the variance. As a result, the Bradley-Terry model is studied. The second model, is parameterized with the same vector $\mathbf{w}$, nevertheless, the probability of any permutation (solution) σ is calculated as

$$P(\sigma|\mathbf{w}) \propto \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \frac{w_{\sigma(i)}}{w_{\sigma(i)} + w_{\sigma(j)}}$$

Despite the similarity with Plackett-Luce, as described in Alza et al. (2018), given the same set of solutions to model, Bradley-Terry defines a probability distribution with a larger variance than that described by Plackett-Luce (see Fig. 1 in Alza et al. (2018)). It is the difference in the variance between the two models that permits the authors to balance the trade-off diversification-intensification by building a mixture model with two components, a Plackett-Luce model and a Bradley Terry model, respectively. Then, the probability of any permutation (solution) is calculated

$$P(\sigma) = \Theta P_{BT}(\sigma) + (1 - \Theta) P_{PL}(\sigma)$$

The Θ parameter is calculated with Nogueira's index (Nogueira and Sechidis 2017), which describes the "diversity" of the modelled solutions. When $\Theta = 0$, the population is highly diverse, and then, the PL model is learnt and sampled. As the algorithm converges, the population becomes more homogeneous ($\Theta \rightarrow 1$), and thus, the BT model is used to sample with a larger variance.

3 Open challenges for future research

In the previous section, we described the milestones of the process of designing an EDA for an optimization problem, and presented the different alternatives/trends that have been published. However, there are strategies that we have not included in the roadmap since, to the best of our knowledge, they have not been extensively investigated yet, and belong to future research lines. In this section, we group them in three blocks: designing new probability models, incorporating problem-specific knowledge and learning from experience.

3.1 Designing new probability models on the search space of solutions

As stated in Stage 1 of the roadmap (see Example 1), in most of the cases, the domain of definition of the probability model does not correspond to Ω, in fact, the model is defined on a set S for which Ω is a subset. In this sense, an alternative is to seek in the statistics and probability theory literature for a probability model that holds the required restrictions, $S = \Omega$. With the exception of a few problem types, in general, this option is not practicable, and thus, it is necessary to design new probability models for the problem at hand. Moreover, it is necessary to provide efficient mechanisms to learn the parameters of the model and to draw samples from it. Illustrative examples are given in Example 2, where a problem defined on a subset of solutions of the binary domain is approached.

An alternative to the option to the previous point is to search for bijective transformations that permit the search space of solutions to be mapped to another space on to which defining probability models is easier. For instance, in the case permutation problems, whose search space is $\mathbb{S}_n$, as for TSP, possible transformations that permit permutations to be mapped to vectors of integers spaces are published in Regnier-Coudert and McCall (2014); Doignon et al. (Mar 2004). In fact, a recent study on the use of bijective mappings for permutation problems in the context of EDAs has obtained promising results (Malagon et al. 2020).

3.2 Incorporating problem-specific knowledge

In addition to the mechanisms that we have enumerated in Stage 3 of the roadmap, an alternative option, as far as we know not yet exploited in EDAs, considers using Bayesian models in order to introduce information about the problem in the form of prior probabilities on the parameters of the model. Under the Bayesian approach, we assume that there is uncertainty about the probability distribution that models the data (solutions from S), and thus, rather than having a specific probability distribution at each generation, we have a (posterior) probability distribution over the parameters. In other words, a probability distribution over probability distributions is defined[5]. This idea is formalized in the equation below

$$P(\boldsymbol{\theta}|D) \propto P(\boldsymbol{\theta}) \prod_{x \in D} P(x|\theta) \tag{5}$$

where θ stands for the set of parameters of the probability model used in the EDA, and D denotes the set of samples used to learn the model. $P(\boldsymbol{\theta})$ denotes the prior probability of $\boldsymbol{\theta}$, i.e., it represents our belief in the set of parameters. The product term in Eq. 5 computes the likelihood of the sample D given the vector of parameters $\boldsymbol{\theta}$. On the basis of the Bayes' theorem, the product of the two terms above is proportional to $P(\theta|D)$, which stands for the probability on the sets of parameters $\boldsymbol{\theta}$ given the data D, also known as the posterior distribution. A sample obtained from the posterior distribution corresponds to a set of parameters that describes a specific probability model.

[5] A summarized introduction to Bayesian statistics can be found in Calvo et al. (2018).

Conversely to the standard EDA scope, under the Bayesian approach, the prior distribution permits our belief in the parameters of the probability model to be defined explicitly. It is this point where the problem specific knowledge can be incorporated after transforming the data to fit in the form of probability of θ parameters. Note that, when compared to other methods for incorporating problem specific knowledge, this approach presents a number of challenges that arise either in the learning or sampling steps. On the one hand, the incorporation of the knowledge is not so direct and requires a strategy to transform the knowledge into prior probabilities on the parameters. On the other hand, the main problem comes from finding a closed form equation of the posterior distribution $P(\boldsymbol{\theta}|D)$, which implies integrating over D. Instead of calculating the exact form of the equation, an alternative option is to sample sets of $\boldsymbol{\theta}$ parameters from the posterior distribution using MCMC-like approximation methods. This option, although feasible, increases the time complexity of the algorithm, and thus, it has to be worth it in terms of overall performance of the algorithm.

In this context, the Bayesian version of a number of known probability models defined on combinatorial domains has already been published (Vitelli et al. 2017; Crispino and Antoniano-Villalobos 2019; Calvo et al. 2018).

3.3 Learning from experience

EDAs are, in general, Markovian nature algorithms. Consequently, the probability model at a given iteration is exclusively determined by the model in the previous iteration, however, the information gained since the first iterations of the optimization is not fully considered. Nonetheless, the path followed by the EDA, described as a sequence of probability models, might provide very useful information, not only about the characterization of good solutions, but also to identify the characteristics that worsen the solutions. Thus, an obvious question arises: *How can we use the experience gained across the iterations to improve the performance of the algorithm?* From the question above, there are other relevant questions that can be formulated:

- How do we characterize the "experience"? Which are the relevant variables to collect during the optimization?
- How do we manage the experience in such a way that permits more efficient EDAs to be implemented?

Looking at the literature on metaheuristics, we find that this idea was pursued decades ago by Glover with the work on Tabu Search (TS) (Glover 1998). The authors realized that not visiting the solutions that have already been evaluated and discarded by the algorithm could be important. Doing so, it would be possible to guide the algorithm towards non-visited areas of the search space. In the initial versions, this idea was implemented by using a memory to store the solutions that the algorithm should avoid. Later, advances on TS designs proposed storing the operations (instead of full solutions) that make the solutions lose quality. However, works in this line have limited advantages as they require the use of memories to store the information (solutions, operations...), and the success of the approach depends on the size of the memory used.

In addition to the usual metaheuristic field ideas, there are a number of inspiring works in the field of machine learning. When it comes to collecting and using all information available for the optimization, Bayesian Optimization (Brochu et al. 2010) is clearly a referent paradigm. As described in Roman et al. (2020), BO is a sequential optimization algorithm, where the sampling strategy is based on a probability distribution over all the possible objective functions. This probability distribution acts as a surrogate model, and it is updated every time a solution is evaluated using the actual objective function. This optimization strategy is interesting when the computation of the objective function is very costly, and thus, the optimization is carried out evaluating few candidates. As a result, the information of all the evaluated solutions is essential to accurately build the surrogate model, and the more solutions that are available, the more accurate the surrogate model is. The success of the BO strategy relies on the suitability of the kernel function and the acquisition function, and their efficient computation.

Recent works on optimization have proposed incorporating procedures and operators that come from the field of Bayesian Optimization to metaheuristic algorithms. Particularly, the authors in Lan et al. (2020); Goff et al. (2020) focused on and tried to exploit the ability of BO to aggregate the information of all the evaluated solutions and accurately sample new solutions. In fact, having a model of the objective function that collects all the evidence that has been created throughout the optimization process is, definitively, an interesting research direction to be considered.

Alternatively, publications in the last few years in the field of deep learning and reinforcement learning have made great advances when achieving competitive results for a number of well known combinatorial problems (Dai et al. 2017; Cappart et al. 2021; Joshi et al. 2020). Although the optimization framework is rather different, and frequently, heuristics are not considered in the experimental comparison of algorithms (for some obscure reason), there are interesting ideas to consider and follow for future works. In fact, works as the one by Bengio et al. (2018) point to the idea of developing algorithms that

combine recent *deep learning* optimization approaches together with classical metaheuristic algorithms. Beyond the straight forward sequential hybridizations, works by Wu et al. (2021) and Chen and Tian (2019) propose interesting lines to consider. In these cases, the proposed optimization algorithms are built on top of classical hill-climbers but they use NN models that collect the information observed throughout the optimization (and also pre-trained knowledge) and guide the steps of the base-algorithm more efficiently. Although works that consider EDAs have not been observed in the literature, similar procedures as in Wu et al. (2021); Chen and Tian (2019) can easily be studied in the future.

4 Conclusions

Estimation of distribution algorithms have gained much popularity in the literature of meta-heuristics for solving either combinatorial or continuous domain optimization problems. Motivated by their robustness, these algorithms have been used to solve a diverse set of real-world and academic problems. Nonetheless, using them implies that the practitioner has knowledge of evolutionary computation and especially of probabilistic modeling. As a result, in many works we see that their usage has deviated from the initial purpose of EDAs.

In this paper, divided in four stages, we presented a roadmap for those practitioners interested in designing EDAs for solving optimization problems. In this roadmap, we addressed the key aspects involved in the design and application of EDAs. Furthermore, in order to illustrate each of the stages, we provided examples from existing works in the literature.

Finally, in addition to the roadmap, future lines for research regarding EDAs were presented. Specifically, and not limited to what follows, notes on Bayesian statistics to develop methods for incorporating information about the problem explicitly in the model, or developing strategies that exploit the information kept in the sequence of models generated across the optimization were discussed.

References

Alden M, Miikkulainen R (2016) MARLEDA: effective distribution estimation through Markov random fields. Theoret Comput Sci 633:4–18

Alza J, Ceberio J, Calvo B (2018) Balancing the diversification-intensification trade-off using mixtures of probability models. In: 2018 IEEE congress on evolutionary computation (CEC), pp 1–8

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

Arza E, Perez A, Irurozki E, Ceberio J (2020) Kernels of mallows models under the hamming distance for solving the quadratic assignment problem. Swarm Evol Comput 59:100740

Ayodele M, McCall J, Regnier-Coudert O, Bowie L (2017) A random key based estimation of distribution algorithm for the permutation flowshop scheduling problem. In: 2017 IEEE congress on evolutionary computation (CEC), pp 2364–2371

Baluja S (2006) Scalable optimization via probabilistic modeling. Studies in computational intelligence, volume 33, chapter incorporating a priori knowledge in probabilistic-model based optimization, pp 205–222. Springer, Berlin

Bengio Y, Lodi A, Prouvost A (2018) Machine learning for combinatorial optimization: a methodological tour d'Horizon. Eur J Oper Res 290():405–421

Bosman PAN, Grahl J (2008) Matching inductive search bias and problem structure in continuous estimation-of-distribution algorithms. Eur J Oper Res 185(3):1246–1264

Bosman PAN, Thierens D (2000) Mixed IDEAs. Technical report, Utrech University

Bosman PAN, Thierens D (2002) Mult-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. Int J Approx Reason 31(3):259–289

Brochu E, Cora VM, De Freitas N (2010) A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. http://arxiv.1012.2599

Brownlee A, Pelikan M, McCall J, Petrovski A (2008) An application of a multivariate estimation of distribution algorithm to cancer chemotherapy. In: Proceedings of the 2008 ACM genetic and evolutionary computation conference, pp 463–464

Calvo B, Ceberio J, Lozano JA (2018) Bayesian inference for algorithm ranking analysis. In: Proceedings of the genetic and evolutionary computation conference companion, GECCO '18, pp 324–325, New York, NY, USA, ACM

Cappart Q, Chételat D, Khalil E, Lodi A, Morris C, Veličković P (2021) Combinatorial optimization and reasoning with graph neural networks

Carnero M, Hernández J, Sánchez M (2018) Optimal sensor location in chemical plants using the estimation of distribution algorithms. Ind Eng Chem Res 57(36):12149–12164

Ceberio J, Irurozki E, Mendiburu A, Lozano JA (2012) A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Progress Artif Intell 1(1):103–117

Ceberio J, Mendiburu A, Lozano JA (2013) The Plackett-Luce ranking model on permutation-based optimization problems. In: 2013 IEEE congress on evolutionary computation, pp 494–501

Ceberio J, Mendiburu A, Lozano JA (2017) A square lattice probability model for optimising the graph partitioning problem. In: 2017 IEEE Congress on evolutionary computation (CEC), pp 1629–1636. IEEE

Ceberio J, Mendiburu A, Lozano JA (2018) Distance-based exponential probability models on constrained combinatorial optimization problems. In: 2018 genetic and evolutionary computation conference (GECCO-2018), Kyoto, Japan, pp 137–138. ACM

Chen X, Tian Y (2019) Learning to perform local rewriting for combinatorial optimization. In: Advances in neural information processing systems (NeurIPS 2019), vol 32. ISBN: 9781713807933.

Crispino M, Antoniano-Villalobos I (2019)Informative extended mallows priors in the bayesian mallows model. ArXiV. arXiv: 1901.10870

Critchlow JVD, Fligner M (1991) Probability models on ranking. J Math Psychol 35:294–318
Dai H, Khalil EB, Zhang Y, Dilkina B, Song B (2017) Learning combinatorial optimization algorithms over graphs. In: Advances in neural information processing systems, vol 2017-Decem, pp 6349–6359
De Bonet JS, Isbell CL, Jr, Viola P (1996) Mimic: finding optima by estimating probability densities. In: Proceedings of the 9th international conference on neural information processing systems, NIPS'96, pp. 424–430, Cambridge, MA, USA, 1996. MIT Press
Doignon J-P, Pekeč A, Regenwetter M (2004) The repeated insertion model for rankings: Missing link between two subset choice models. Psychometrika 69(1):33–54
Echegoyen C, Lozano J, Santana R, Larranaga P (2007) Exact bayesian network learning in estimation of distribution algorithms. pp 1051–1058
Echegoyen C, Mendiburu A, Santana R, Lozano JA (2012) Toward understanding edas based on bayesian networks through a quantitative analysis. IEEE Trans Evol Comput 16(2):173–189
Echegoyen C, Mendiburu A, Santana R, Lozano JA (2013) On the taxonomy of optimization problems under estimation of distribution algorithms. Evol Comput 21(3):471–495
Etxeberria R, Larrañaga P (1999) Global optimization with bayesian networks. In: II symposium on artificial intelligence, special session on distributions and evolutionary optimization, CIMAF99, pp 332–339
Fard MR, Mohaymany AS (2019) A copula-based estimation of distribution algorithm for calibration of microscopic traffic models. Transp Res Part C Emerg Technol 98:449–470
Fligner MA, Verducci JS (1986) Distance based ranking Models. J R Stat Soc 48(3):359–369
Gallagher M (2000) Multi-layer perceptron error surfaces: visualization, structure and modelling models for iterative global optimization. PhD thesis, Queensland University
Glover FLM (1998) Handbook of combinatorial optimization, chapter Tabu search. Springer, Berlin
Goff L, Buchanan E, Hart E, Eiben A, Li W, de Carlo M, Hale M, Angus M, Woolley R, Timmis J, Winfield A, Tyrrell A (2020) Sample and time efficient policy learning with cma-es and bayesian optimisation. pp 432–440
Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison/Wesley, Reading MA
Goldberg DE, Lingle R (1985) Alleles, Loci and the traveling salesman problem. In: ICGA, pp 154–159
Hauschild M, Pelikan M, Sastry K, Goldberg D (2011) Using previous models to bias structural learning in the hierarchical boa. Evol Comput 20:135–160
Höns R (2012) Using maximum entropy and generalized belief propagation in estimation of distribution algorithms. In: Shakya S, Santana R (editors) Markov networks in evolutionary computation. Springer, pp 175–190
Irurozki E (2014) Sampling and learning distance-based probability models for permutation spaces. PhD thesis, University of the Basque Country
Irurozki E, Ceberio J, Santamaria J, Santana R, Mendiburu A (2018) Algorithm 989: Perm_mateda: a matlab toolbox of estimation of distribution algorithms for permutation-based combinatorial optimization problems. ACM Trans Math Softw 44(4):47:1-47:13
Iyer PVK (1950) The theory of probability distributions of points on a lattice. Ann Math Stat 21(2):198–217
Jiang S, Ziver A, Carter J, Pain C, Goddard A, Franklin S, Phillips H (2006) Estimation of distribution algorithms for nuclear reactor fuel management optimisation. Ann Nucl Energy 33(11–12):1039–1057
Joshi CK, Cappart Q, Rousseau L-M, Laurent T, Bresson X (2020) Learning TSP requires rethinking generalization. pp 1–22
Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
Kollat JB, Reed PM, Kasprzyk JR (2008) A new epsilon-dominance hierarchical bayesian optimization algorithm for large multi-objective monitoring network design problems. Adv Water Resour 31(5):828–845
Krejca M, Witt C (2018) Theory of estimation-of-distribution algorithms. CoRR, abs/1806.05392
Lan G, Tomczak J, Roijers D, Eiben A (2020) Time efficiency in optimization with a bayesian-evolutionary algorithm
Lange K (2010) Numerical analysis for statisticians. Springer, Berlin
Larrañaga P, Lozano JA (2002) Estimation of distribution algorithms: a new tool for evolutionary computation. Kluwer Academic Publishers, New York
Lebanon G, Mao Y (2008) Non-parametric modeling of partially ranked data. J Mach Learn Res (JMLR) 9:2401–2429
Lima CF, Pelikan M, Lobo FG, Goldberg DE (2009) Engineering stochastic local search algorithms. designing, implementing and analyzing effective heuristics, chapter loopy substructural local search for the Bayesian optimization algorithm. Springer, Berlin Heidelberg, pp 61–75
Lozano JA, Larrañaga P, Inza I, Bengoetxea E (2006) Towards a new evolutionary computation: advances on estimation of distribution algorithms (studies in fuzziness and soft computing). Springer, New York
Lozano JA, Mendiburu A (2002) Solving job schedulling with estimation of distribution algorithms. In: Larrañaga P, Lozano JA (eds) Estimation of distribution algorithms. A new tool for evolutionary computation, pp. 231–242. Kluwer Academic Publishers, New York
Malagon M, Irurozki E, Ceberio J (2020) Alternative representations for codifying solutions in permutation-based problems. In: 2020 IEEE congress on evolutionary computation (CEC), pp 1–8
Marden JI (1996) Analyzing and modeling rank data. CRC Press
Mendiburu A, Santana R, Lozano JA (2012) Fast fitness improvements in estimation of distribution algorithms using belief propagation. In: Santana R, Shakya S (eds) Markov networks in evolutionary computation. Springer, Berlin, pp 141–155
Mezuman E, Weiss Y (2012) Globally optimizing graph partitioning problems using message passing. In: Proceedings of the 15th international conference on artificial intelligence and statistics (AISTATS), pp 770–778
Mühlenbein H (1998) The equation for response to selection and its use for prediction. Evol Comput 5:303–346
Mühlenbein H, Mahnig T (2002) Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. Int J Approx Reason 31(3):157–192
Mühlenbein H, Mahnig T (1999) FDA—a scalable evolutionary algorithm for the optimization of additively decomposed functions. Evol Comput 7(4):353–376
Mühlenbein H, Paaß G (1996) From recombination of genes to the estimation of distributions I. Binary parameters. In: Lecture notes in computer science 1411: parallel problem solving from nature—PPSN IV, pp 178–187
Murphy TB, Martin D (2003) Mixtures of distance-based models for ranking data. Comput Stat Data Anal 41(3–4):645–655
Nogueira BGS, Sechidis K (2017) On the use of spearman's rho to measure the stability of feature rankings. In: Alexandre RJL, Salvador Sánchez J (eds) Pattern recognition and image analysis. IbPRIA 2017, vol 10255. Springer
Pelikan M, Goldberg DE, Lobo FG (2002) A survey of optimization by building and using probabilistic models. Comput Optim Appl 21(1):5–20

Pelikan M, Sastry K, Cantú-Paz E (2006) Scalable optimization via probabilistic modeling: from algorithms to applications (studies in computational intelligence). Springer, New York

Peña JM, Lozano JA, Larrañaga P (2005) Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks. Evolut Comput, pp 43–66

Regnier-Coudert O, McCall J (2014) Factoradic representation for permutation optimisation. In: Bartz-Beielstein T, Branke J, Filipič B, Smith J (eds) Parallel problem solving from nature—PPSN XIII. Springer, pp 332–341

Roman I, Mendiburu A, Santana R, Lozano JA (2020) Bayesian optimization approaches for massively multi-modal problems. In: Matsatsinis NF, Marinakis Y, Pardalos P (eds) Learning and intelligent optimization. Springer, Berlin, pp 383–397

Santana R, Bielza C, Larranaga P, Lozano JA, Echegoyen C, Mendiburu A, Armananzas R, Shakya S (2010) Mateda-2.0: estimation of distribution algorithms in matlab. J Stat Softw 35(7):1–30

Santana R, Larrañaga P, Lozano JA (2007) Challenges and open problems in discrete edas. Technical report, Department of Computer Science and Artificial Intelligence, University of the Basque Country

Santana R, Larrañaga P, Lozano JA (2008) Protein folding in simplified models with estimation of distribution algorithms. IEEE Trans Evol Comput 12:418–438

Santana R, Mendiburu A, Zaitlen N, Eskin E, Lozano JA (2010) Multi-marker tagging single nucleotide polymorphism selection using estimation of distribution algorithms. Artif Intell Med 50(3):193–201

Schwarz J, Ocenasek J (2000) A problem knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning. In: Proceedings of the 4th joint conference on knowledge-based software engineering. IOS Press, pp 51-58

Shakya S, McCall J (2007) Optimization by estimation of distribution with DEUM framework based on Markov random fields. Int J Autom Comput 4(3):262–272

Shakya S, Santana R (2012) Markov networks in evolutionary computation. Springer, Berlin

Shapiro JL (2005) Drift and scaling in estimation of distribution algorithms. Evol Comput 13(1):99–123

Soto M, Gonzalez-Fernandez Y, Ochoa-Zezzatti C (2015) Modeling with copulas and vines in estimation of distribution algorithms. Inves Oper 36:1–23

Thurstone L (1927) A law of comparative judgment. Psychol Rev 34:273–286

Vitelli V, Sørensen Ø, Crispino M, Frigessi A, Arjas E (2017) Probabilistic preference learning with the mallows rank model. J Mach Learn Res 18(1):5796–5844

Wang C, Ma H, Chen G, Hartmann S (2018) Towards fully automated semantic web service composition based on estimation of distribution algorithm. In: Mitrovic T, Xue B, Li X (eds) AI 2018: advances in artificial intelligence. Springer, Cham, pp 458–471

Wright AH, Pulavarty S (2005) Estimation of distribution algorithm based on linkage discovery and factorization. In: 2007 genetic and evolutionary computation conference (GECCO-2005), Washington D.C., USA. ACM, pp 695–703

Wu Y, Song W, Cao Z, Zhang J, Lim A (2021) Learning improvement heuristics for solving routing problems. IEEE Trans Neural Netw Learn Syst, pp 1-13

Zhang Q, Muhlenbein H (2004) On the convergence of a class of estimation of distribution algorithms. IEEE Trans Evol Comput 8(2):127–136