

# Learning search distributions in estimation of distribution algorithms with minimalist diffusion models

**Abstract**—Estimation of Distribution Algorithms (EDAs) are evolutionary algorithms that model the distribution of the best solutions in order to bias the search to promising areas of the search space. The interplay between the accuracy and efficiency of the models when learning and sampling search distributions are key aspects for their success. Recently, a variety of generative models have been investigated within the context of EDA applications. In this paper, we propose the use of diffusion models as a way to represent and exploit the distribution of selected solutions. We first discuss on the suitability of diffusion models currently applied in machine learning for their application to EDAs, identifying the strengths and weaknesses of these methods. Then, we propose a way to incorporate a feasible diffusion model approach as part of EDAs and evaluate different variants of this approach on the recently introduced GNBG benchmark of continuous optimization functions.

**Index Terms**—estimation of distribution algorithms, diffusion models, evolutionary algorithms, alpha-deblending, search distributions

## I. INTRODUCTION

Although different research groups converged on the introduction and formalization of Estimation of Distribution Algorithms (EDAs) [1]–[3] around 30 years ago, three of the common reasons that motivated the research on these algorithms were: (1) The need to address some characteristics of genetic algorithms (GAs) [4], [5] that significantly harm their performance, in particular, the building-block disruption problem; (2) The desire for a more precise mathematical characterization and theoretical analysis of GAs; and (3) The quest for unveiling the connections between GAs and methodological developments from other research areas (e.g., statistical physics, probabilistic graphical models, etc). These research directions remain relevant nowadays.

One key aspect that has been among the most treated questions in EDA research since its origins, is how to find feasible ways to model the distributions of the best solutions. In its general definition, the question is simple, how to model the distribution of the high-fitness (selected) solutions in such a way that the models could be used to generate similarly high-fitness solutions, and in this way move the search to more promising regions. While posing the question is straightforward, finding the right answer is complicated due to the many criteria that should be taken into account at the time of choosing an effective modeling approach: Firstly, the model should have the necessary capacity (or expressivity) as to guarantee that key relationships between the variables of the problems, usually expressed as variable dependencies, are

captured. Secondly, the learning method should be able to learn from a relatively small set of solutions and in a short time, since it is usually applied in each EDA generation. Finally, sampling methods to be used with the learned model should be also affordable in terms of the computational time, and be able to generate high-fitness solutions similar, but not identical to those used for learning the model.

A large variety of model classes have been since the early days of EDA research, from simple univariate models that assume independence among the variables [6]–[8], to tree-based models that generally provide a good trade-off between complexity and accuracy of the models [2], [9]–[11], to more complex probabilistic models able to capture intricate multivariate interactions such as Bayesian networks [12]–[14] and Markov networks [15], [16]. In the continuous domain, most developments have been focused on Gaussian models [17], [18], although other types of models have also been investigated [19]–[21].

Generative methods based on neural networks have revolutionized machine learning (ML) applications [22], [23]. Since these methods can be used to generate samples from a latent representation of probability distributions, they have been also investigated in EDAs. In [24], generative adversarial networks (GANs) were applied for modeling EDA search distributions within the context of single-objective optimization [24], [25]. Variational autoencoders (VAEs) were also proposed as the basis for learning and sampling method for EDAs that solve discrete optimization problems [26], [27]. The dilemma between the accuracy of sampling and the computational cost needed to learn and sample models is exacerbated when generative approaches based on neural networks are used since these methods, in particular those that require the synchronization of different submodels such as GANs, can be significantly costly in terms of time.

Diffusion models (DMs) [28], [29] have emerged as one of the most effective classes of generative models in a variety of application domains. They operate by gradually transforming noise into structured samples through a reverse diffusion process. They typically consist of two phases: a *forward process*, where data is progressively corrupted by adding noise over multiple steps, and a *reverse process*, where a neural network is trained to denoise the data, thereby reconstructing the original distribution. By iteratively refining noisy inputs, diffusion models can generate high-quality samples that closely resemble real data.

The well-known success of diffusion models lies in their ability to effectively model complex probability distributions. However, since neural networks play a pivotal role in DM, as they are essential for learning complex probability distributions and enabling efficient sampling, the processes of learning the neural networks and generating solutions from them can be computationally costly, and convergence is not always guaranteed. This poses the question of whether and under which adaptations can DMs be used as part of iterative optimizers such as EDAs where learning and sampling the DM is expected to occur several times. Some clues to answer this question may come from other domains of applications where DMs have been applied. Although DM has been primarily used for image generation, they have been shown to be very effective in other domains such as natural language processing [30], bioinformatics [31], and climate science [32].

The emergence of DMs opens a number of research directions for EDAs. The first is, whether these methods can be more efficient at the time of modeling the search distributions. Then, whether it is possible to find a path from random distributions, or other initial distributions to the target distribution of high quality solutions which are modeled in EDAs. Also, which information could this path linking different distributions provide about the characteristics of the optimization problem and the solutions search space. We argue that it is not only the potentially more accurate results provided by DMs what could be relevant for EDAs, but the understanding of the process of transforming the distributions itself. Finally, another important question is whether the use of DM is feasible in terms of the computational time needed to learn these models.

In this paper, we introduce and investigate DMs [29], [33] as models to learn and sample search distributions in EDAs. Since computational complexity and accuracy are two critical issues of application of DM, we research which class of DMs are appropriate for their application within EDAs, and propose the adaptations needed in order to make them work efficiently as part of EDAs. We introduce the Diffusion-by-Deblending family of EDA (DbD-EDAs). The algorithms in this family are based on a low-cost minimalist deterministic diffusion model called  $\alpha$ -(de)Blending [34]. We introduce different members of this family by using different ways to define the initial and target distributions that relate the forward and reverse processes in DMs. The performance of the algorithms is evaluated on a set of functions constructed using the Generalized and Configurable Benchmark Generator (GNBG) for continuous numerical optimization [35]. This benchmark allows to model different facets of difficulty for EAs, including the amount and strength of interactions between variables.

The paper is organized as follows: Section II provides a brief introduction to EDAs, including a discussion of the sampling methods commonly used in this domain. Section III introduces diffusion models and their mathematical formalization, including the explanation of the  $\alpha$ -(de)Blending method. DbD-EDA is introduced in Section IV. Section V reviews related studies in the intersection between EAs and diffusion models, and their contributions. The experimental benchmark and the

results of our experiments are presented in Section VI. Finally, Section VII concludes the paper and discusses potential directions for future research.

## II. ESTIMATION OF DISTRIBUTION ALGORITHMS

EDAs are a class of evolutionary algorithms that use probabilistic models to guide the search process, replacing traditional genetic operators such as crossover and mutation. EDAs learn a probabilistic model from a set of selected, promising solutions, and sample new candidate solutions from this model. The rationale behind this approach is that the probabilistic model captures characteristic patterns of the promising solutions by encoding relevant statistics about problem variables and their dependencies. Explicitly modeling these relationships is particularly advantageous for problems with strong dependencies. This represents a fundamental difference from GAs, which implicitly exploit these relationships through the selection and recombination of building blocks.

### Algorithm 1: Estimation of Distribution Algorithm

---

```

1  Set  $t \leftarrow 0$ . Generate  $N$  solutions randomly.
2  do {
3    Evaluate the solutions using the fitness function.
4    Select a population  $D_t^S$  of  $K \leq N$  solutions
      according to a selection method.
5    Calculate a probabilistic model of  $D_t^S$ .
6    Generate  $N$  new solutions by sampling from the
      distribution represented in the model.
7    Combine the current population with the new solu-
      tions and replace the current population.
8     $t \leftarrow t + 1$ 
9  } until Termination criteria are met.
```

---

Algorithm 1 outlines the steps of a typical EDA. The algorithm begins by generating an initial population of solutions, typically at random. In some cases, a specific heuristic or seeding method may be employed if prior knowledge about the approximate location of optimal solutions is available. The solutions are then evaluated using a fitness function, and a subset of the best solutions is selected based on their fitness values. Different selection methods can be applied, and, as in other EAs, solutions with better fitness values have a higher probability of being selected.

Model building and sampling are the two defining steps of EDAs. In the model-building step, a probabilistic model is constructed from the selected solutions. This model captures the statistical relationships and dependencies between the problem variables. In the sampling step, new solutions are generated by sampling from the learned probabilistic model. This step replaces the traditional genetic operators used in GAs. The rationale is that the most salient features of the best solutions, as captured by the probabilistic model, will be preserved in the newly generated solutions.

As in other EAs, the replacement step combines the solutions from the previous generation with the current set

of sampled solutions. This ensures that the best solutions found are retained, contributing to maintaining diversity in the population. The steps of selection, model building, and sampling are repeated for a number of generations until a termination criterion is met. Termination criteria may include reaching a maximum number of generations, achieving a predefined solution quality, or other conditions.

#### A. Model Building in EDAs

Solutions selected according to fitness are likely to exhibit a different genotypic distribution than the whole population from which they were selected. It is expected that selected solutions retain patterns and traits that it is convenient to exploit in order to generate other high-fitness solutions. Modeling this distribution pursues the goal of capturing these patterns in terms of statistical regularities which will be present both in the marginal, and depending on the class of models, also in the structure of interactions and conditional dependencies.

A distinguishing characteristic of EDAs is the type of probabilistic model used. The choice of model depends on the variable representation and the problem being addressed. In terms of representation, there are classes of probabilistic graphical models specifically suited for discrete representations, such as Bayesian networks, and others designed for continuous variables, such as Gaussian networks. EDAs have also been proposed for problems with mixed representations, although problems with this type of representation have received less attention in the literature.

In terms of complexity, more intricate models can capture more complex dependencies but may also be computationally expensive to learn and sample from. Simpler models, such as univariate marginal models, are computationally efficient but may not be suitable for problems with strong dependencies.

#### B. Learning and sampling in EDAs

The choice of the probabilistic model and its ability to capture relevant information about promising solutions cannot be separated from the decision about the sampling method used to generate new solutions. Sampling methods determine how accurately the information contained in the models is transferred to the new population. They can positively bias the search toward promising areas of the search space while also maintaining diversity. Among the most commonly employed sampling methods in EDAs are Probabilistic Logic Sampling (PLS), Gibbs Sampling (GS), Belief Propagation (BP), sampling methods with generative NNs, and sampling methods used with non-generative NNs. We briefly review these methods in order to highlight later the specificities of DMs, in particular when used as sampling methods.

PLS is the most widely used sampling method in EDAs [36]. It generates new solutions by sampling each variable in the graphical model according to its conditional probability distribution, given the values of its parent variables. Variables are sampled in topological order, ensuring that parents are sampled before their children. While PLS is efficient, it requires an initial ordering of variables, which may ignore

certain dependencies in the graph. GS iteratively samples each variable conditioned on the current values of all other variables in the model. This allows for the representation of cyclic or higher-order dependencies. However, GS is computationally expensive, especially for multimodal problems, as convergence may require many iterations [37].

BP is an inference method in which the nodes in the graphical model exchange messages to calculate marginal probabilities. When the algorithm converges, max-marginals (for max-product) or marginal functions (for sum-product) are obtained. In the max-product approach, each variable in the optimal solution is assigned the value with the highest probability at each max-marginal. This max-marginal variant of BP has been used as a way to compute the  $k$  most probable solutions according to the model, which are assumed to be the top ranked high-fitness solutions. However, BP can be also computationally costly and in order to work effectively it should be combined with PLS [38].

When neural networks are used as models in EDAs a variety of sampling methods can be applied. For typical NN generators such as GANs and VAEs, the corresponding methods for these models are used. GANs sample from a generator that has been trained to produce solutions that are indistinguishable from the training set (high quality solutions in this case), VAEs sample from a latent space, usually a multivariate Gaussian distribution, and transform these samples to the original problem representation (solution representation for EDAs).

There are also sampling methods used to generate solution from neural networks that are not generators. Among them is *Artificial Neural Network (ANN) Inversion* (Back-drive) [39]. Given a desired output value of a neural network, Back-drive determines the input values that would produce that output. This is applied in EDAs to sample from a multilayer perceptron (MLP) by setting the desired output (high fitness value) and inferring a genotype whose evaluation is likely to match the entered fitness.

In addition to Back-drive, sampling methods that use adversarial perturbations [40] can generate high-fitness solutions by modifying a random or low-quality solution until the classification given by a surrogate neural network classifier is changed from the low-fitness class to the high-fitness class of solutions. The rationale is that the adversarial perturbation will indeed lead to adding to the initial solution patterns that according to the model are characteristic of high-fitness solutions.

### III. DIFFUSION MODELS

#### A. Denoising diffusion models

Denoising diffusion models operate by progressively adding noise to the original data in a forward process, gradually transforming the data distribution into a pure noise distribution. Then, in a backward process a model learns how to reverse this process to generate samples from the underlying data distribution. This approach is based on a Markovian noising

process and a learned denoising process, which we describe in detail below following the original papers [28], [29], [33].

1) *Forward Process (Diffusion Process)*: The forward process, also known as the diffusion process, gradually adds noise (typically Gaussian noise) to a sample  $x_0 \sim q(x_0)$  over  $T$  time steps. This process is defined as a Markov chain, where each step perturbs the data according to a variance schedule.

At each time step  $t$ , noise is added as follows:

$$q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I), \quad (1)$$

where:

- $x_0$  is a sample from the real data distribution  $q(x_0)$ ,
- $\beta_t \in (0, 1)$  is a variance schedule that controls the amount of noise added at step  $t$ ,
- $I$  is the identity matrix,
- $q(x_t | x_{t-1})$  is a transition probability (Gaussian in this case).

$x_t$  can be expressed directly as a function of  $x_0$  and Gaussian noise  $\epsilon$  []:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \quad (2)$$

where:

$$\alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{s=1}^t \alpha_s. \quad (3)$$

This formulation allows us to sample  $x_t$  directly from  $x_0$  without iterating through intermediate steps.

2) *Reverse Process (Denoising Process)*: In the backward process, a denoising neural network is trained to reverse the forward process by learning to remove the noise added to  $x_0$  in the forward process. The model predicts the noise that was added at each step and progressively refines the sample. Therefore, the objective of the reverse process is to learn to denoise a sample  $x_t$  to recover  $x_0$ . When the forward process is a Gaussian transition, the true reverse transition is also Gaussian:

$$q(x_{t-1} | x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I), \quad (4)$$

where:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}x_t, \quad (5)$$

$$\tilde{\beta}_t = \frac{(1 - \bar{\alpha}_{t-1})\beta_t}{1 - \bar{\alpha}_t}. \quad (6)$$

Since  $x_0$  is unknown, a neural network is trained to predict the noise  $\epsilon$  in  $x_t$  so that  $x_0$  could be estimated. The neural network  $\epsilon_\theta(x_t, t)$  is trained to minimize the denoising loss:

$$\mathbb{E}_{x_0, \epsilon, t} [\|\epsilon - \epsilon_\theta(x_t, t)\|^2]. \quad (7)$$

Using this learned noise prediction, the estimated mean for the denoising step is:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right), \quad (8)$$

which allows sampling from the learned reverse process:

$$x_{t-1} = \mu_\theta(x_t, t) + \sqrt{\tilde{\beta}_t}z, \quad z \sim \mathcal{N}(0, I). \quad (9)$$

## B. Limitations of denoising DMs

While the application of diffusion models to learning the distribution of the promising solutions and generating new solutions in EDAs can be seen as a natural extension of their remarkable success in accurately sampling from complex distributions, we argue that translating these techniques to EDAs requires careful consideration. In particular, reducing the computational time associated to the number of steps needed for effectively learning the path between random and target distributions, and doing sampling in the reverse process, is critical for an effective implementation of DM within EDAs.

Active research is being conducted to reduce the number of steps required for generating samples from diffusion models [33], [41], [42]. Among the methods that are being considered for this purpose are: replacing the backward stochastic differential equation with an ordinary differential equation [41], which is then solved using a highly efficient algorithm; and using the Denoising Diffusion Implicit Model (DDIM) [33], a non-Markovian diffusion process that traverses only a subset of latent variables, allowing for faster sampling by reducing the number of denoising steps required.

An important consideration for EDA applications is whether the intermediate samples generated during the forward and backward processes—particularly those with low noise levels—could be used to further explore the search space. This could be feasible for optimization problems where the evaluation of the objective function is not computationally expensive.

## IV. DIFFUSION-BY-DEBLENDING (DBD) EDA

Since the key element for the application of DMs to EDAs is the computational time, we have opted for using a variant of DM that is very efficient. The learning iterative alpha-deblending algorithm [34] is inspired in algorithms used for deblending images and, remarkably, it allows to start the forward process from samples that do not follow a random uniform distribution. This is relevant for EDAs since we can reuse solutions already evaluated in order to accelerate the path linking process toward the optimal solutions.

In addition to the choice of the DM model, the Diffusion-by-Deblending (DbD) EDA introduced in this paper incorporates a new restart mechanism that complements the exploitative nature of the DM. Restart methods have been extensively used in EAs [43] since they are an effective way to scape local optima and make a more efficient exploration of the search space. Therefore we split the presentation of DbD-EDA to introduce the following components of the algorithm:

- The alpha-deblending model.
- Alpha-deblending learning and sampling algorithms.
- Choice of the initial distribution.
- Restart mechanism.

### A. Alpha-deblending DB

The alpha deblending method involves training a neural network  $D$  to predict the average posterior samples. The training objective is to minimize the expected  $l_2$  norm between the neural network's prediction and the average difference vector

between posterior samples. The original learning objective of the method is defined as:

$$\min_{\Theta} \mathbb{E}_{\alpha, x} \left[ \left\| D_{\Theta}(x, \alpha) - \mathbb{E}_{(x_0, x_1) | (x, \alpha)} [x_1 - x_0] \right\|^2 \right] \quad (10)$$

where:

- $D_{\Theta}$  is the neural network with parameters  $\Theta$ .
- $x$  is a blended sample.
- $\alpha$  is the blending parameter.
- $(x_0, x_1)$  are random samples from densities  $p_0$  and  $p_1$ , respectively.
- The expression  $\mathbb{E}_{(x_0, x_1) | (x, \alpha)} [x_1 - x_0]$  represents the average difference vector between the posterior samples, given  $x$  and  $\alpha$ .

Minimizing the  $l_2$  norm of the average of a distribution is equivalent to minimizing the  $l_2$  norm of all samples of the distribution. The equivalent objective is:

$$\min_{\Theta} \mathbb{E}_{\alpha, x, (x_0, x_1) | (x, \alpha)} \left[ \left\| D_{\Theta}(x, \alpha) - (x_1 - x_0) \right\|^2 \right] \quad (11)$$

Sampling  $x \sim p_{\alpha}$  first and then  $(x_0, x_1) | (x, \alpha)$  is equivalent to sampling  $(x_0, x_1) \sim (p_0, p_1)$  and blending them to obtain  $x \sim p_{\alpha}$ . This leads to the final learning objective:

$$\min_{\Theta} \mathbb{E}_{\alpha, x_0, x_1} \left[ \left\| D_{\Theta}((1 - \alpha)x_0 + \alpha x_1, \alpha) - (x_1 - x_0) \right\|^2 \right] \quad (12)$$

where:

- $x = (1 - \alpha)x_0 + \alpha x_1$  is a blended sample of  $x_0$  and  $x_1$ .
- The neural network  $D_{\Theta}$  is trained to predict the average difference vector  $(x_1 - x_0)$  given the blended sample  $x$  and the blending parameter  $\alpha$ .

In practice, [SEE variant (d) of Table 1 is used], where the neural network is trained to predict the average difference vector between the posterior samples.

### B. Learning and sampling methods

#### Algorithm 2: Training

---

```

1 Require  $x_0 \sim p_0, x_1 \sim p_1, \alpha \sim U$ 
2   #Compute the blended sample
3    $x = (1 - \alpha)x_0 + \alpha x_1$ .
4   #Calculate the loss
5    $L = \|D_{\Theta}(x, \alpha) - (x_1 - x_0)\|^2$ .
6   Backpropagate from  $L$  and update the parameters  $\Theta$ .
```

---

The pseudocode of the learning method is shown in Algorithm 2. The training algorithm requires samples  $x_0$  and  $x_1$  from densities  $p_0$  and  $p_1$  respectively, and a blending parameter  $\alpha$  sampled from a uniform distribution between 0 and 1. A blended sample  $x$  is computed as a linear interpolation of  $x_0$  and  $x_1$  using  $\alpha$ . The loss  $L$  is calculated as the squared Euclidean norm (L2 norm) of the difference between the neural network's prediction  $D_{\Theta}(x, \alpha)$  and the difference vector  $(x_1 - x_0)$  as described in the previous section. The algorithm then performs backpropagation to update the parameters  $\Theta$  of the neural network, minimizing the loss  $L$ .

#### Algorithm 3: Sampling

---

```

1 Require  $x_0 \sim p_0, T, \alpha := \frac{t}{T}$ 
2 for  $t = 0$  to  $t = T - 1$ 
3   # Update the sample
4    $x_{\alpha_{t+1}} = x_{\alpha_t} + (\alpha_{t+1} - \alpha_t) D_{\Theta}(x_{\alpha_t}, \alpha_t)$ .
```

---

The pseudocode of the sampling method is shown in Algorithm 3. The algorithm requires an initial sample  $x_0$  from the density  $p_0$ , the total number of iterations  $T$ , and a blending schedule defined as  $\alpha := \frac{t}{T}$ . It iteratively updates the sample  $x$  from  $x_{\alpha_t}$  to  $x_{\alpha_{t+1}}$  for  $t$  from 0 to  $T - 1$ . In each iteration, the sample is updated by adding the product of the difference in blending parameters  $(\alpha_{t+1} - \alpha_t)$  and the neural network's prediction of the average posterior difference  $D_{\Theta}(x_{\alpha_t}, \alpha_t)$ .

#### C. DbD-EDA variants: Choice of the initial distribution

The alpha-deblending method, and in general, DMs require an original and target distributions in order to learn the forward and reverse diffusion paths. In most of applications, the target distribution is clearly defined. When applied as part of EDAs, both the target and source distributions can be selected in order to prioritize some aspect of the search.

In this paper we propose to consider four different choices that intend to capture a different perspective of what the search for the optimal solutions is about. Each choice of the source and target distributions defines an DbD-EDA variant. We have defined the following variants:

- 1) DbD-CS:  $p_0$  is formed by solutions sampled from the  $\{C\}$ urrent population and  $p_1$  comprises solutions sampled from the  $\{S\}$ elected population.
- 2) DbD-CD: (Selected solution distribution target)  $p_0$  is formed by solutions sampled from the  $\{C\}$ urrent population and  $p_1$  is constructed conditioned on  $p_0$ . For each solution  $x^i$  in  $p_0$ ,  $\hat{x}^i$  in  $p_1$  contains the solution in the selected population that is closest to  $x^i$  in terms of the mean squared error (MSE)  $\{D\}$ istance.
- 3) DbD-UC:  $p_0$  is formed by solutions sampled from a  $\{U\}$ nivariate approximation of the current population ( $p_c^u$ ) and  $p_1$  is sampled from the  $\{C\}$ urrent population.
- 4) DbD-US:  $p_0$  is formed by solutions sampled from a  $\{U\}$ nivariate approximation of the current population ( $p_c^u$ ) and  $p_1$  is sampled from the  $\{S\}$ elected population.

Each of the DbD-EDA variants can use its own initial distribution during the sampling step. DbD-CS, DbD-CD, and DbD-UC start from the selected distribution. DbD-US starts from samples of a univariate approximation of the selected distribution.

DbD-CS is the method that resembles the most the goal of DM, it tries to learn the distribution of the selected solution and uses as original distribution the current solutions. The method can be directly related to truncated DMs [?]. It is assumed, that it will be faster to transit between solutions in the population (which were generated in previous generations) and the high-fitness selected solutions. At the time of sampling,

starting from selected solutions intends to improve the fitness of these solutions, going beyond the quality of the solutions already explored. DbD-CD follows a similar strategy, but in this case the target solutions are filtered in order to make the transition between each pair of initial and target solutions faster.

DbD-UC implements a different objective. In this case, by learning the univariate approximation of the current population, we intend to disrupt the possible interactions that exist between the variables. Then, the goal of the DM is to learn how to recover these interactions by making the transit between the samples from the univariate approximation and the current solutions. DbD-US is similar to DbD-UC, but in this case it is assumed that the DM will learn the interactions only if they are required for an accurate approximation of the selected solutions.

#### D. Restart strategy

Preliminary experiments with the DbD-EDAs showed that the algorithms tend to converge to local optima after a few iterations due to their increased exploitation capabilities. This behavior is not unique of DbD-EDAs since of the EDAs such as the Gaussian UMDA exhibit a similar pattern. Therefore, we designed a restart mechanism that could be triggered when the algorithm is stucked in a local minimum.

### V. RELATED WORK

#### A. Diffusion models for optimization

One promising domain for the application of diffusion models is optimization, particularly black-box (BB) optimization. Additionally, there is growing interest in using diffusion models to find optimal policies in reinforcement learning [44].

A limited number of recent works have proposed the use of diffusion models for black-box (BB) optimization, where the objective function can be computationally expensive. In this context, the goal of the diffusion model is to generate increasingly better solutions with improved objective values until the optimal solution is found. One approach to BB optimization using diffusion models is the Denoising Diffusion Optimization Models (DDOM) [45], which learns a conditional generative model over the domain of the BB function, conditioned on the function values, given an offline dataset. This approach aligns with previous applications of neural networks as surrogate functions [44].

In [46], the authors propose integrating diffusion models as a component of an expensive multi-objective Bayesian optimization algorithm (MOBO). Bayesian optimization is well-suited for this scenario, and various variants have been proposed for multi-objective optimization problems. The authors of [46] use the diffusion model to learn the distribution of the Pareto set. The dimensionality of the problems considered was relatively low (up to 10 decision variables), and the authors acknowledge that the relatively high computational cost of the algorithm makes it challenging to apply to very high-dimensional problems.

Yan and Jin [47] introduce *EmoDM*, an evolutionary multi-objective algorithm that incorporates a diffusion process component. The algorithm is closer to a transfer learning approach than to a typical evolutionary optimizer. The diffusion model is trained on a set of source multi-objective problems and is subsequently applied to solve a target problem. A distinctive feature of this approach is that the noise generated in the forward pass is derived from the evolution of an EA for the source instances. This noise corresponds to the differences between solutions in two consecutive populations, with the diffusion process proceeding in reverse order relative to the evolution. Specifically, the diffusion process starts from the last (converged) population and adds noise until it reaches the initial random population. The approach introduced in [47] also conditions the decision variables on the objective values. While this approach demonstrates the benefits of diffusion models within an evolutionary framework, it relies on the transfer of information among problem instances and is therefore fundamentally different from the EDA scenarios considered in this paper.

In [48], the authors propose integrating diffusion models as an offspring-generation process in EAs. Similar to an EDA, the idea is to sample solutions from high-fitness regions in the parameter space. As in [47], both decision variables and objectives are used to learn the model. A distinctive feature of this approach is the introduction of a weighting function in the loss function to bias the model toward high-fitness samples. The proposed algorithm is tested on two toy problems (double peaks and Rastrigin) and the cart-pole reinforcement learning problem. The authors conclude that the introduced method produces significant improvements in adaptability to changing environments while maintaining reliable convergence to target solutions.

#### B. Bridging and modeling search distributions in EDAs

Significant developments in EDAs are based on the use of the Boltzmann distribution to define an ideal selection probability, where solutions with better fitness receive an exponentially higher probability [49]. Early convergence results for EDAs were based on the Boltzmann EDA (BEDA) [50], an EDA with an infinite population where solutions were selected using the Boltzmann distribution.

### VI. EXPERIMENTS

#### A. GNBG benchmark

GNBG is a powerful and versatile benchmarking framework designed for evaluating global optimization algorithms on single-objective, box-constrained, continuous numerical problems. Unlike traditional test suites that rely on a fixed collection of standard benchmark functions, GNBG introduces a single, parametric baseline function that can be flexibly configured to generate a wide variety of problem landscapes. This design empowers researchers to tailor problem instances with precisely controllable characteristics, enabling targeted analysis of algorithmic behavior. Among the facets of difficulty

that GNBG considers are: Modality (unimodal to highly multimodal), Ruggedness, Deceptiveness, Symmetry and asymmetry, Separability and variable interaction structures, Dimensionality and conditioning, and Basin shape and morphology.

We have used the benchmark suite composition proposed for the the competition challenge of CEC-2025. It comprises 24 functions categorized into:

- Unimodal functions (f1–f6)
- Single-component multimodal functions (f7–f15)
- Multi-component multimodal functions (f16–f24)

## B. Experimental results

## VII. CONCLUSIONS

We discuss the benefits and limitations of using diffusion models compared to traditional sampling methods employed in EDAs. This paper also identifies critical research areas necessary for the successful integration of diffusion models into EAs, as well as the current obstacles that must be overcome to enhance the efficiency of these algorithms.

This position paper has explored the integration of diffusion models into evolutionary optimization, with a particular focus on EDAs. We have argued that diffusion models, with their ability to accurately model complex probability distributions, offer a promising alternative to traditional probabilistic models used in EDAs. By analyzing the potential advantages and limitations of diffusion-based sampling, we have highlighted their suitability for problems with high-dimensional, multimodal, or discrete representations. Furthermore, we have identified critical research challenges, such as the need for efficient sampling methods, the adaptation of diffusion models for discrete spaces, and the theoretical analysis of their convergence properties. These insights provide a foundation for future work aimed at enhancing the performance and applicability of EDAs through the integration of diffusion models.

A key contribution of this paper is the identification of three main areas of focus for the successful application of diffusion models in EDAs: the characteristics of the optimization problems, the properties of the sampling process, and the theoretical underpinnings of diffusion models. We have discussed how the flexibility of diffusion models, particularly in guided versus unguided generation and the approximation of score functions, can be leveraged to improve the quality and diversity of solutions in EDAs. Additionally, we have emphasized the importance of reducing the computational cost of sampling and the potential of combining diffusion models with other sampling techniques, such as message-passing algorithms, to address high-dimensional problems.

Finally, this paper has underscored the need for further theoretical and empirical research to fully realize the potential of diffusion models in evolutionary optimization. By bridging the gap between diffusion models and EDAs, we aim to inspire new directions in optimization research, ultimately leading to more robust and efficient algorithms capable of tackling complex real-world problems. Future work should focus on addressing the identified challenges, implementing

and exploring hybrid approaches, and validating the proposed methods across a broader range of applications.

## REFERENCES

- [1] H. Mühlenbein and G. Paaß, “From recombination of genes to the estimation of distributions I. Binary parameters,” in *Parallel Problem Solving from Nature - PPSN IV*, ser. Lectures Notes in Computer Science, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., vol. 1141. Berlin: Springer, 1996, pp. 178–187.
- [2] S. Baluja and S. Davies, “Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space,” in *Proceedings of the 14th International Conference on Machine Learning*, D. H. Fisher, Ed. San Francisco, CA: Morgan Kaufmann, 1997, pp. 30–38.
- [3] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, “Optimization by learning and simulation of Bayesian and Gaussian networks,” Department of Computer Science and Artificial Intelligence, University of the Basque Country, Technical Report EHU-KZAA-IK-4/99, 1999.
- [4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [5] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.
- [6] C. González, J. A. Lozano, and P. Larrañaga, “Mathematical modeling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions,” *International Journal of Approximate Reasoning*, vol. 31, no. 4, pp. 313–340, 2002.
- [7] Z. G. Arenas, J. C. Jimenez, L.-V. Lozada-Chang, and R. Santana, “Estimation of distribution algorithms for the computation of innovation estimators of diffusion processes,” *Mathematics and Computers in Simulation*, vol. 187, pp. 449–467, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0378475421000926>
- [8] C. González, J. A. Lozano, and P. Larrañaga, “Analyzing the population based incremental learning algorithm by means of discrete dynamical systems,” *Complex Systems*, vol. 12, no. 4, pp. 465–479, 2001.
- [9] M. Pelikan and H. Mühlenbein, “Marginal distributions in evolutionary algorithms,” in *Proceedings of the International Conference on Genetic Algorithms Mendel ’98*, Brno, Czech Republic, 1998, pp. 90–95.
- [10] A. Ochoa, M. R. Soto, R. Santana, J. Madera, and N. Jorge, “The factorized distribution algorithm and the junction tree: A learning perspective,” in *Proceedings of the Second Symposium on Artificial Intelligence (CIMA-99)*, A. Ochoa, M. R. Soto, and R. Santana, Eds. Havana, Cuba: Editorial Academia. Havana, Cuba, March 1999, pp. 368–377.
- [11] R. Santana, A. Ochoa, and M. R. Soto, “The mixture of trees factorized distribution algorithm,” in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshek, M. Garzon, and E. Burke, Eds. San Francisco, CA: Morgan Kaufmann Publishers, 2001, pp. 543–550.
- [12] R. Etxeberria and P. Larrañaga, “Global optimization using Bayesian networks,” in *Proceedings of the Second Symposium on Artificial Intelligence (CIMA-99)*, A. Ochoa, M. R. Soto, and R. Santana, Eds. Editorial Academia. Havana, Cuba, 1999, pp. 332–339.
- [13] H. Mühlenbein and T. Mahnig, “FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions,” *Evolutionary Computation*, vol. 7, no. 4, pp. 353–376, 1999.
- [14] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, “BOA: The Bayesian optimization algorithm,” in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. I. Orlando, FL: Morgan Kaufmann Publishers, San Francisco, CA, 1999, pp. 525–532.
- [15] R. Santana, “A Markov network based factorized distribution algorithm for optimization,” in *Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003)*, ser. Lecture Notes in Artificial Intelligence, vol. 2837. Dubrovnik, Croatia: Springer, 2003, pp. 337–348. [Online]. Available: <http://dx.doi.org/10.1007/b13633>
- [16] S. Shakyia and J. McCall, “Optimization by estimation of distribution with DEUM framework based on Markov random fields,” *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 262–272, 2007.

- [17] E. Bengoetxea, T. Miquélez, P. Larrañaga, and J. A. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2002, ch. Experimental results in function optimization with EDAs in continuous domain, pp. 177–190.
- [18] P. A. Bosman and D. Thierens, “Expanding from discrete to continuous estimation of distribution algorithms: The IDEA,” in *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*. Paris, France: Springer, September 16–20 2000, lecture Notes in Computer Science 1917.
- [19] —, “IDEAs based on the normal kernels probability density function,” Utrecht University, Tech. Rep. UU-CS-2000-11, 2000.
- [20] M. Gallagher, M. Frean, and T. Downs, “Real-valued evolutionary optimization using a flexible probability density estimator,” in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, vol. I. Orlando, FL: Morgan Kaufmann Publishers, San Francisco, CA, 1999, pp. 840–846.
- [21] P. Pošík, “BBOB-benchmarking a simple estimation of distribution algorithm with Cauchy distribution,” in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2009*. New York, NY, USA: ACM, 2009, pp. 2309–2314.
- [22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [23] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” *CoRR*, vol. abs/1312.6114, 2013. [Online]. Available: <http://arxiv.org/abs/1312.6114>
- [24] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, “Evolutionary multiobjective optimization driven by generative adversarial networks (GANs),” *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3129–3142, 2020.
- [25] M. Probst, “Generative adversarial networks in estimation of distribution algorithms for combinatorial optimization,” *CoRR*, vol. abs/1509.09235, 2015. [Online]. Available: <http://arxiv.org/abs/1509.09235>
- [26] S. Bhattacharjee and R. Gras, “Estimation of distribution using population queue based variational autoencoders,” in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 1406–1414.
- [27] U. Garcíarena, R. Santana, and A. Mendiburu, “Expanding variational autoencoders for learning and exploiting latent representations in search distributions,” in *Proceedings of the 2018 on Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 849–856. [Online]. Available: <https://dl.acm.org/doi/10.1145/3205455.3205645>
- [28] M. Chen, S. Mei, J. Fan, and M. Wang, “An overview of diffusion models: Applications, guided generation, statistical rates and optimization,” *CoRR*, vol. abs/2404.07771, 2024. [Online]. Available: <http://arxiv.org/abs/2404.07771v1>
- [29] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.
- [30] H. Zou, Z. M. Kim, and D. Kang, “A survey of diffusion models in natural language processing,” *CoRR*, vol. abs/2305.14671, 2023. [Online]. Available: <http://arxiv.org/abs/2305.14671>
- [31] A. Schneuing, C. Harris, Y. Du, K. Didi, A. Jamasb, I. Igashov, W. Du, C. Gomes, T. L. Blundell, P. Lio *et al.*, “Structure-based drug design with equivariant diffusion models,” *Nature Computational Science*, vol. 4, no. 12, pp. 899–909, 2024.
- [32] L. Li, R. Carver, I. Lopez-Gomez, F. Sha, and J. Anderson, “Generative emulation of weather forecast ensembles with diffusion models,” *Science Advances*, vol. 10, no. 13, p. eadk4489, 2024.
- [33] J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” *CoRR*, vol. abs/2010.02502, 2020. [Online]. Available: <http://arxiv.org/abs/2010.02502>
- [34] E. Heitz, L. Belcour, and T. Chambon, “Iterative  $\alpha$ -(de) blending: A minimalist deterministic diffusion model,” in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023, pp. 1–8.
- [35] D. Yazdani, M. N. Omidvar, D. Yazdani, K. Deb, and A. H. Gandomi, “GNBG: A generalized and configurable benchmark generator for continuous numerical optimization,” *CoRR*, vol. abs/2312.07083, 2023. [Online]. Available: <http://arxiv.org/abs/2312.07083>
- [36] P. Larrañaga and C. Bielza, “Estimation of distribution algorithms in machine learning: A survey,” *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 5, pp. 1301–1321, 2024.
- [37] J. A. Gámez, J. L. Mateo, and J. M. Puerta, “EDNA: Estimation of dependency networks algorithm,” in *Bio-inspired Modeling of Cognitive Tasks, Second International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2007*, ser. Lecture Notes in Computer Science, J. Mira and J. R. Álvarez, Eds., vol. 4527. Springer, 2007, pp. 427–436.
- [38] R. Santana, A. Mendiburu, and J. A. Lozano, “A review of message passing algorithms in estimation of distribution algorithms,” *Natural Computing*, vol. 15, no. 1, pp. 165–180, 2016. [Online]. Available: <https://link.springer.com/article/10.1007/s11047-014-9473-2>
- [39] S. Baluja, “Learning deep models of optimization landscapes,” in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–10.
- [40] U. Garcíarena, J. Vellido, A. Mendiburu, and R. Santana, “Adversarial perturbations for evolutionary optimization,” in *International Conference on Machine Learning, Optimization, and Data Science (LOD-2021)*, ser. Lecture Notes in Computer Science, vol. 13164. Springer, 2021, pp. 408–422.
- [41] C. Lu, S. Li, and Z. Lu, “Building energy prediction using artificial neural networks: A literature survey,” *Energy and Buildings*, vol. 262, p. 111718, 2022.
- [42] Y. Ren, H. Chen, Y. Zhu, W. Guo, Y. Chen, G. M. Rotskoff, M. Tao, and L. Ying, “Fast solvers for discrete diffusion models: Theory and applications of high-order algorithms,” *CoRR*, vol. abs/2502.00234, 2025. [Online]. Available: <http://arxiv.org/abs/2502.00234>
- [43] A. Auger and N. Hansen, “A restart CMA evolution strategy with increasing population size,” in *Proceedings of The IEEE Congress on Evolutionary Computation (CEC-2005)*, vol. 2. IEEE, 2005, pp. 1769–1776.
- [44] H. Zheng, P. He, W. Chen, and M. Zhou, “Truncated diffusion probabilistic models,” *CoRR*, vol. abs/2202.09671, 2022. [Online]. Available: <http://arxiv.org/abs/2202.09671>
- [45] Z. Zhu, H. Zhao, H. He, Y. Zhong, S. Zhang, H. Guo, T. Chen, and W. Zhang, “Diffusion models for reinforcement learning: A survey universal adversarial audio perturbations,” *CoRR*, vol. abs/2311.01223, 2024. [Online]. Available: <http://arxiv.org/abs/2311.01223v4>
- [46] S. Krishnamoorthy, S. Mashkaria, and A. Grover, “Diffusion models for black-box optimization,” in *Proceedings of the 40th International Conference on Machine Learning*, Honolulu, Hawaii, 2023, p. 1–16.
- [47] B. Li, Z. Di, Y. Lu, H. Qian, F. Wang, P. Yang, K. Tang, and A. Zhou, “Expensive multi-objective Bayesian optimization based on diffusion models,” *CoRR*, vol. abs/2405.08674, 2024. [Online]. Available: <http://arxiv.org/abs/2405.08674>
- [48] X. Yan and Y. Jin, “EmoDM: A diffusion model for evolutionary multi-objective optimization,” *CoRR*, vol. abs/2401.15931, 2024. [Online]. Available: <http://arxiv.org/abs/2401.15931>
- [49] B. Hartl, Y. Zhang, H. Hazan, and M. Levin, “Heuristically adaptive diffusion-model evolutionary strategy,” *CoRR*, vol. abs/2411.13420, 2024. [Online]. Available: <http://arxiv.org/abs/2411.13420>
- [50] T. Mahnig and H. Mühlenbein, “Comparing the adaptive Boltzmann selection schedule SDS to truncation selection,” in *Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)*, Havana, Cuba, March 2001, pp. 121–128.
- [51] H. Mühlenbein, T. Mahnig, and A. Ochoa, “Schemata, distributions and graphical models in evolutionary optimization,” *Journal of Heuristics*, vol. 5, no. 2, pp. 213–247, 1999.