Semiparametric Estimation of Distribution Algorithms for Continuous Optimization

Vicente P. Soloviev, Concha Bielza, *Senior Member, IEEE*, and Pedro Larrañaga, *Senior Member, IEEE*

***Abstract*—Traditional estimation of distribution algorithms (EDAs) often use Gaussian densities to optimize continuous functions, such as the estimation of Gaussian network algorithms (EGNAs) which use Gaussian Bayesian networks (GBNs). However, this assumes a parametric density function, and, in GBNs, linear dependencies between variables. Furthermore, the EGNA baseline learns a GBN at each iteration based on the best individuals in the last iteration, which may lead to local optimum convergence or large variance between solutions across multiple independent runs of the algorithm. In this work, we propose a semiparametric EDA in which the restriction of assuming Gaussianity in the variables is relaxed using semiparametric Bayesian networks (SPBNs), in which nodes estimated by kernels coexist with nodes that assume Gaussianity, and the algorithm itself is able to determine where to use each type of node. Additionally, our approach takes into account information from several past iterations to learn the SPBN from which the new solutions are sampled in each iteration. The empirical results show that semiparametric EDAs are a useful tool for continuous scenarios compared to different kinds of EDAs and other optimization techniques in continuous environments.**

***Index Terms*—Benchmarking, continuous optimization, estimation of distribution algorithms (EDAs), semiparametric Bayesian network (SPBN).**

I. Introduction

ESTIMATION of distribution algorithms (EDAs) [1] have been broadly studied in recent decades in the context of continuous optimization [2], [3], [4]. They belong to the family of evolutionary algorithms (EAs) [5], where the main difference among the different members is the way in which new solutions are generated during runtime. EAs, and more specifically, EDAs, have attracted much attention and interest due to their successful application and their easy implementation in different optimization tasks [6], [7], [8]. In EDAs, compared to traditional EAs, such as genetic algorithms [9], new solutions are not generated through crossover and mutation operators, but are sampled from a probabilistic model. The probabilistic model is iteratively updated with the best individuals selected from previous generations of the algorithm.

Most continuous real-world optimization problems can be formulated as a cost function to be minimized, $\min g(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{R}^n$ represents the value assignment of a set of continuous random variables. In recent years, the automation of processes in industry has increased the need to optimize certain tasks that involve continuous variables, either given some data generated by sensors or cost functions. In the literature, we can find different probabilistic models embedded in EDAs being used to approach continuous optimization problems, such as the estimation of Gaussian network algorithm (EGNA) [10], due to their simplicity and speed.

However, such models have certain disadvantages due to the use of parametric probability distributions. First, assuming Gaussianity when the search space of the optimization problem is a continuous environment that does not fit a Gaussian distribution, may result in poor solutions. Second, the use of Gaussian distributions during the runtime of the algorithm tends to shrink the search space represented by the standard deviation or the covariance matrix as the iterations of the algorithm progress, which can result in premature convergence, or large variance in the results found in different runs [11]. Third, some algorithms, such as EGNA, use probabilistic models that consider dependencies between the variables; assuming Gaussianity in this type of model also implies assuming linear dependencies between variables, which has consequences for the way in which the algorithm navigates the landscape.

To try to address these shortcomings, some works use nonparametric models to avoid assuming a specific probability distribution over the search space, such as the univariate kernel EDA (KEDA) [12], in which a univariate kernel density estimation (KDE) is used for modeling each variable as a unique probabilistic model of the EDA. Despite improving the results over some state-of-the-art EDAs, this approach does not consider KDE multivariate probability models, or models in which only some of the variables fit a parametric probability distribution.

In this article, we propose a new variant of EDAs, for optimizing problems in continuous multivariate environments, in which we take advantage of the benefits provided by Gaussian Bayesian networks (GBNs), and the accuracy of KDE-based models to find a tradeoff between accuracy and computational cost. Our approach embeds a semiparametric Bayesian network (SPBN) [13] in which KDE variables coexist with Gaussian variables in the same probabilistic graphical model, where it is the algorithm itself that decides in each generation

Manuscript received 3 November 2022; revised 3 March 2023 and 16 May 2023; accepted 27 June 2023. Date of publication 29 June 2023; date of current version 1 August 2024. This work was supported in part by the Spanish Ministry of Science and Innovation under Project PID2019-109247GB-I00, Project TED2021-131310B-I00, and Project RTC2019-006871-7. The work of Vicente P. Soloviev was supported by the Predoctoral Grant FPI from the Spanish Ministry of Science and Innovation under Grant PRE2020-094828. *(Corresponding author: Vicente P. Soloviev.)*

The authors are with the Computational Intelligence Group, Universidad Politécnica de Madrid, 28660 Madrid, Spain (e-mail: vicente.perez.soloviev@fi.upm.es; mcbielza@fi.upm.es; pedro.larranaga@fi.upm.es).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2023.3290670.

Digital Object Identifier 10.1109/TEVC.2023.3290670

which variables take which type of probability distribution throughout the runtime. During the runtime, if the algorithm detects that all the variables are Gaussian in some iteration (extreme case), the proposed approach will use a GBN. Thus, we believe that our algorithm is able to identify the promising areas of the search space due to the use of SPBNs but is a less heavy computational approach than using only KDEs. Furthermore, a recurrent problem in the use of EDAs is the iterative variance reduction that can lead to premature convergence [8]. Therefore, we have designed an archive-based approach in which the probabilistic model is updated not only with the best individuals of the previous generation but also with the best individuals of several previous generations.

The remainder of this article is organized as follows. Section II reviews the theoretical background on Bayesian networks (BNs), and their parametric, nonparametric, and semiparametric variants for continuous environments. Section III explains the baseline of EDAs for continuous multivariate spaces and the different variants available in the state-of-the-art methods. Section IV introduces the new semiparametric EDA we propose. Section V reports the experimental results after comparing our approach to some state-of-the-art optimizers and analyzes their CPU time and complexity. Section VI ends this article with some conclusions and further work.

II. Bayesian Networks

BNs [14] are a type of probabilistic graphical model that compactly represent the joint probability distribution of a set of random variables. BNs are widely used in machine learning [15] for different applications [16], [17] due to their capability of representing the uncertain knowledge contained in the data and the possibility of adding expertise.

BNs can be defined as a pair $(G, \boldsymbol{\Theta})$ over a set of random variables $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$. They are composed of: 1) a directed acyclic graph (DAG) $G = \{V, A\}$, where V denotes the variables in $\boldsymbol{X}$ represented as nodes in the DAG and A contains the arcs between the nodes, which encode the conditional (in)dependence relationships among the variables and 2) a set of parameters $\boldsymbol{\Theta}$ that define the conditional probability distribution (CPD) of each variable X_i given its parents $\boldsymbol{Pa}_i$ in G, where the parents of a variable X_i are the variables directly pointing at X_i in the DAG.

Considering this notation, the joint probability distribution $P(\boldsymbol{X})$ over a set of random variables $\boldsymbol{X}$ is defined as a product of all the CPDs of each variable given its parents $(P(X_i|\boldsymbol{Pa}_i))$ in the graph

$$P(\boldsymbol{X}) = P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\boldsymbol{Pa}_i).$$

When dealing with discrete variables, the CPD is a tabular representation, where $P(X_i|\boldsymbol{Pa}_i)$ is encoded as a table in which each entry is a joint assignment to X_i and each variable in $\boldsymbol{Pa}_i$. Note that each entry in the CPD table is non-negative and is restricted to $\sum_{x_i \in \Omega(X_i)} P(x_i|\boldsymbol{pa}_i) = 1$, where $\boldsymbol{pa}_i$ denotes a value assignment for $\boldsymbol{Pa}_i$, and $\Omega(X_i)$ refers to the domain of X_i.

However, when dealing with continuous variables, a tabular representation is unfeasible, as each variable assumes a continuous density function. The variables may be discretized, but this can lead to poor results on some occasions. To improve the model fitting, more have complex solutions included assuming parametric or nonparametric distributions, which led to parametric BNs and nonparametric BNs, respectively. The combination of both parametric and nonparametric distributions in a BN led to the newer SPBNs [13].

A. Parametric BNs for Continuous Variables

When a BN assumes a parametric distribution for each of its variables, such as Gaussian, then the BN is a GBN, where all CPDs are defined using a linear Gaussian CPD

$$f(X_i|\boldsymbol{pa}_i) = \mathcal{N}\left(\beta_{i0} + \beta_{i1} pa_{i_1} + \cdots + \beta_{ik} pa_{i_k}; \sigma_i^2\right) \quad (1)$$

where $\beta_{i1}, \ldots, \beta_{ik}$ are the weights associated with each of the parents of X_i, σ_i^2 is the variance, and β_{i0} is the intercept.

GBNs are the most widely used models in the BN area when working with continuous data, as they provide many advantages, such as ease of implementation, the speed of fitting a dataset to a Gaussian distribution, and the existence of closed formulas for performing inference in such models.

However, when the data do not fit Gaussians, then the distribution is poorly modeled. Moreover, as GBNs assume linear interactions between X_i and $\boldsymbol{Pa}_i$, they are not applicable in representing nonlinear relationships among variables.

Other approaches related to the assumption of probabilistic distributions include mixtures of multivariate Gaussian distributions [18], mixtures of Gaussians [19], mixtures of truncated exponentials [20], mixtures of polynomials [21], and mixtures of truncated basis functions [22]. However, because of factors, such as the existence of closed formulas for inference and the fact that learning the parameters is less expensive, the usage of GBNs is more flexible and widespread than these alternatives.

B. Nonparametric BNs for Continuous Variables

The alternative to assuming a distribution over a dataset is to use nonparametric models. Nonparametric models have the advantage of better fitting the functions distribution that do not fit parametric models, but they sacrifice speed and simplicity.

An example of this type of nonparametric model is the KDE [23], which may be considered a mixture model in which the number of components is equal to the number of data samples. Consequently, the KDE demands much memory, and this demand grows according to the size of the data used for its training. The KDE joint probability density is defined as

$$f(\boldsymbol{x}) = \frac{1}{N} \sum_{j=1}^{N} K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{x}_j) \quad (2)$$

where $K(\cdot)$ is the n-variate kernel function, $K_{\boldsymbol{H}}(\boldsymbol{x}) = |\boldsymbol{H}|^{-1/2} K(\boldsymbol{H}^{-1/2}\boldsymbol{x})$ is the kernel function, $\boldsymbol{x}_j$ is the jth sample among the N samples used to train the KDE and $\boldsymbol{H}$ is the bandwidth matrix: a square $n \times n$ matrix that defines properties of the KDE such as the smoothness of the density estimation. If a Gaussian kernel is used, then

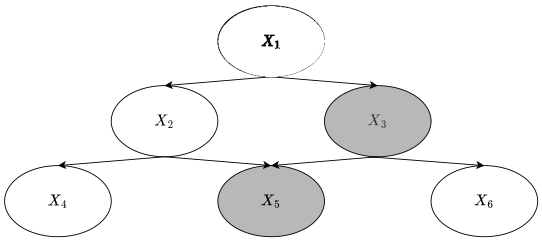


Fig. 1. Structure of an SPBN example with six nodes. The white and gray nodes represent the nodes that have been fitted using Gaussians and KDEs, respectively.

$K(\boldsymbol{x}) = (1/(2\pi)^{n/2})\exp(-(1/2)\boldsymbol{x}^T\boldsymbol{x})$, where $\boldsymbol{x}^T$ is the transpose vector of $\boldsymbol{x}$.

A BN where all the CPDs are estimated with KDEs is named a KDE-based BN (KDEBN). This type of model is less common than GBNs but is still widely used in [24] and [25] because of the goodness of fit provided by the KDEs used. However, KDEBNs inherit the disadvantages of KDEs, such as a high memory demand that grows linearly with N, the parameters (such as $\boldsymbol{H}$) to be tuned, and the complexity of these models.

Other approaches that avoid parametric models include the combination of GBNs with nonparametric Bayesian mixture models [26] and the use of Gaussian processes to learn functional relations between variables [27]. However, the use of KDEs is more widespread due to the existence of different types of easily implementable kernels.

C. Semiparametric BNs for Continuous Variables

The combination of both nonparametric and parametric CPDs results in SPBNs [13]. This type of model considers the possibility that, nodes that assume a Gaussian distribution and nodes using a KDE can co-exist in the same BN, and that both types of nodes can be connected in the BN structure. Fig. 1 shows an example of an SPBN structure where all possible dependencies between both types of nodes can be found. The seminal paper [13] proposed different algorithms for learning the parameters and the structure of SPBNs. During the learning process these learning algorithms decide between the following two possibilities: the dependency between a variable X_i an its parents $\boldsymbol{Pa}_i$ is linear Gaussian

$$f(X_i|\boldsymbol{Pa}_i) = \mathcal{N}\left(\beta_{i0} + \sum_{X_j \in \boldsymbol{Pa}_i} \beta_{ij} X_j, \sigma_i^2\right)$$

where $\beta_{i1}, \ldots, \beta_{ik}$ are the weights associated with each of the k parents of X_i, β_{i0} is the intercept, and σ_i^2 is the variance of X_i; or the dependency is given by a conditional KDE (CKDE) CPD

$$f(X_i|\boldsymbol{Pa}_i) = \frac{f(X_i, \boldsymbol{Pa}_i)}{f(\boldsymbol{Pa}_i)}$$

where $f(X_i, \boldsymbol{Pa}_i)$ and $f(\boldsymbol{Pa}_i)$ are KDE models as defined in (2).

Note that if all the CPDs in the SPBN are linear Gaussian, then the learned SPBN is equivalent to a GBN, and if all the CPDs in the SPBN are CKDEs, the learned SPBN is equivalent to a KDEBN. For a more exhaustive mathematical formalization, see [13].

1) Parameter Learning: A standard approach for parameter estimation is employing the maximum-likelihood criterion, which aims to select the set of parameters that maximizes the likelihood function. The likelihood function is defined as the probability of the training dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ given the BN model

$$f(\mathcal{D}|\boldsymbol{\Theta}, G) = \prod_{j=1}^{N} f(\boldsymbol{x}_j|\boldsymbol{\Theta}, G) = \prod_{j=1}^{N}\prod_{i=1}^{n} f(x_{ji}|\boldsymbol{\Theta}_i, G) \quad (3)$$

where $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jn})$ represents the jth sample of the dataset $\mathcal{D}$, and $\boldsymbol{\Theta}_i$ is the set of parameters for the CPD of node i. Commonly, the log of the likelihood (log-likelihood) is optimized, as it is considered to provide better numerical precision, and it is defined as

$$\mathcal{L}(G, \Theta : \mathcal{D}) = \sum_{j=1}^{N}\sum_{i=1}^{n} \log f(x_{ji}|\boldsymbol{\Theta}_i, G). \quad (4)$$

For cases in which the variables and the relationships between them are defined as linear Gaussian (1), the maximum-likelihood estimation is obtained using an ordinary least squares estimator [28].

The CKDE CPDs are composed of two nonparametric distributions: 1) $f(X_i, \boldsymbol{Pa}_i)$ and 2) $f(\boldsymbol{Pa}_i)$, which involve the estimation of bandwidth matrices $\boldsymbol{H}_i$ [for $f(x_i, \boldsymbol{Pa}_i)$] and $\boldsymbol{H}_i^-$ [for $f(\boldsymbol{Pa}_i)$]. However, a CKDE CPD can be fitted by estimating only the bandwidth matrix $\boldsymbol{H}_i$ [13]. For the estimation of $\boldsymbol{H}_i$, and due to the impossibility of using maximum-likelihood estimation, the KDE models are trained using other error criterion models, such as the normal reference rule [13], [29], widely used for Gaussian KDE models where the mean integrated squared error is minimized. It defines $\boldsymbol{H}_i = N^{-2/(|\boldsymbol{Pa}_i|+5)}\hat{\boldsymbol{\Sigma}}$, where $\hat{\boldsymbol{\Sigma}}$ is the sample covariance matrix of random variables X_i and $\boldsymbol{Pa}_i$.

2) Structure Learning: The structure of BNs can be learned by three types of methods: 1) score-based algorithms, which attempt to optimize a cost function that evaluates how well a structure fits a dataset; 2) constraint-based algorithms, which perform conditional independence statistical tests to find the optimal structure; or 3) a combination of both. The original paper on SPBNs [13] proposes one algorithm for each score-based and constraint-based structure learning method.

As a score-based algorithm, a modification of the traditional hill climbing (HC) algorithm [30] is proposed. HC is a greedy algorithm in which in each step applies operators ($\mathcal{O}$) over the DAG which involve adding, reversing, or removing arcs in the BN structure. The HC adapted to SPBN change the set of operators, being possible in each step, not only to modify the arcs but also the type of nodes (Gaussian or CKDE). Thus, if all operators are used (adding, reversing or removing arcs, and swapping between both node types) the algorithm iteratively proposes a new SPBN structure and node type configuration. If the node type operator is not used, then the algorithm is equivalent to the classical HC. The resultant structure and node type configuration is the one which optimizes the score (5), where

the parameters of the Gaussian and CKDE CPDs are considered. Schwarz [31] determined that optimizing traditional scores such as the Bayesian information criterion (BIC) leads to overfitting by adding too many arcs in the structure, so they propose using the K-fold cross-validated log likelihood over the training set $\mathcal{D}_{trn}$ as the score to be optimized by the greedy algorithm

$$\mathcal{S}_{CV}^{K}(\mathcal{D}, G) = \sum_{m=1}^{K} \mathcal{L}\left(G, \boldsymbol{\Theta}_{\mathcal{I}_{trn}^{m}} : \mathcal{D}_{\mathcal{I}_{\text{test}}^{m}}\right) \tag{5}$$

where $\mathcal{L}(G, \boldsymbol{\Theta}_{\mathcal{I}_{trn}^{m}} : \mathcal{D}_{\mathcal{I}_{\text{test}}^{m}})$ is the log likelihood (4) of the mth test fold dataset element in an SPBN composed of parameters $\Theta_{\mathcal{I}_{trn}^{m}}$ and DAG G, K is the number of folds for cross validation, and $\mathcal{I}$ refers to the disjoint sets of indices used in the cross-validation technique.

The overfitting is controlled using the validation set $\mathcal{D}_{val} = \mathcal{D} \backslash \mathcal{D}_{trn}$ which measures the goodness of fit of the new structure at each iteration

$$\mathcal{S}_{val}(\mathcal{D}_{trn}, \mathcal{D}_{val}, G) = \mathcal{L}\left(G, \boldsymbol{\Theta}_{\mathcal{D}_{trn}} : \mathcal{D}_{val}\right) \tag{6}$$

where $\boldsymbol{\Theta}_{\mathcal{D}_{trn}}$ are the parameters estimated using $\mathcal{D}_{trn}$.

The HC proposed is improved by using a tabu list to constrain the search space and explore different directions to escape from local optima. Its pseudocode is presented in Algorithm 1. Lines 8–16 describe the process of searching for new structures that aims to maximize the score function (5) by applying different operators from $\mathcal{O}$, and Lines 17–24 enable and disable the tabu list depending on the evaluation of (6). The tabu list prevents the algorithm from applying operators that may undo operations that were recently applied. The algorithm uses the hyperparameter λ as the stopping criterion. λ denotes the number of iterations with no improvement in the best cost found, and once it is reached, the algorithm is considered to have converged.

As it is a constraint-based algorithm, Colombo and Maathuis [32] proposed a modification of the traditional parents–children (PC) algorithm. The traditional PC algorithm performs conditional independence tests, which typically assume that the variables fit a multivariate Gaussian. Runge [33] also proposed using nonparametric tests such as the conditional mutual information k nearest neighbors (CMIknn) test, which estimates mutual information with the k-nearest neighbors, and the randomized conditional correlation test (RCoT) [34], which is a less-resource-demanding test. These two tests are used to find the structure of the SPBN.

The node type of the SPBN is found by using a modified version of the HC algorithm previously explained (Algorithm 1), but in defining the operator set allowing only to modify the node types and not the arcs in the graph already found by the PC algorithm.

3) Asymptotic Complexity: In this section we analyze the complexity of learning GBNs, SPBNs, and KDEBNs.

SPBN and KDEBN learning involves a cross-validated score function (5), which has complexity $\mathcal{O}(KT)$, where K is the number of folds of the cross validation and T is the cost of learning the parameters and evaluating the log likelihood for each fold. However, to evaluate the scores of linear Gaussians or CKDE CPDs is not as complex [13]. To compute the local score of a variable X_i with parents $\boldsymbol{Pa}_i$, the former complexity (for Gaussians) is $\mathcal{O}(kJ|\boldsymbol{Pa}_i|^2)$, where J is the number of training samples on each fold and K is the number of folds, and the latter complexity (for CKDE CPDs) is $\mathcal{O}(NJ|\boldsymbol{Pa}_i|^2)$ where N is the total number of samples. Then, the HC algorithm for SPBN complexity is $\mathcal{O}(Oi\lambda KT)$, where i is the number of tabu lists reinitialized, $i = 1$ is the best of the cases, $O = O_{\text{SPBN}}$ the size of the set of operators defined for the HC, and λ is the patience in Algorithm 1.

Thus, considering the SPBN complexity, learning a KDEBN implies that T will always cost the CKDE learning, $\mathcal{O}(NJ|\boldsymbol{Pa}_i|^2)$, which is more expensive than learning a Gaussian CKDE, and $O = O_{\text{SPBN}} - 1$, as the node type operator is not further needed. In the worst case, the SPBN will learn all the nodes as CKDE, and then, it will be more expensive to learn the SPBN than a KDEBN.

In the case of GBNs, the score function used during the learning process has a complexity of $\mathcal{O}(T)$, where T is the cost of learning the parameters and evaluating the log-likelihood (4). The complexity of the HC algorithm for GBN is $\mathcal{O}(Oi\lambda T)$, where $O = O_{\text{SPBN}} - 1$, as the node type operator is not needed.

Algorithm 1 Greedy Hill-Climbing for SPBNs

Input: Training data $\mathcal{D}$, starting structure G_0, set of operators $\mathcal{O}$, patience λ, number of folds K

Output: Optimal structure G_{best}

1: $G_{best} \leftarrow G_0$
2: $G_{new} \leftarrow G_0$
3: $i \leftarrow 0$
4: $Tabu \leftarrow \emptyset$
5: $\mathcal{D}_{trn}, \mathcal{D}_{val} \leftarrow$ Split($\mathcal{D}$) # training and validation
6: **while** $i < \lambda$ **do**
7: $\quad G \leftarrow G_{new}$
8: $\quad$ **for** o in $\mathcal{O}$ **do**
9: $\quad\quad$ **if** o does not reverse $o' \in$ Tabu **then**
10: $\quad\quad\quad G_{candidate} \leftarrow o(G)$
11: $\quad\quad\quad$ **if** $\mathcal{S}_{CV}^{k}(\mathcal{D}_{trn}, G_{candidate}) > \mathcal{S}_{CV}^{k}(\mathcal{D}_{trn}, G_{new})$ **and** $\mathcal{S}_{CV}^{k}(\mathcal{D}_{trn}, G_{candidate}) - \mathcal{S}_{CV}^{k}(\mathcal{D}_{trn}, G) > 0$ **then**
12: $\quad\quad\quad\quad o_{new} \leftarrow o$
13: $\quad\quad\quad\quad _{new} \leftarrow \mathcal{G}_{candidate}$
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: $\quad$ **if** $\mathcal{S}_{val}(\mathcal{D}_{trn}, \mathcal{D}_{val}, G_{new}) > \mathcal{S}_{val}(\mathcal{D}_{trn}, \mathcal{D}_{val}, G_{best})$ **then**
18: $\quad\quad \mathcal{G}_{best} \leftarrow \mathcal{G}_{new}$
19: $\quad\quad$ Tabu $\leftarrow \emptyset$
20: $\quad\quad i \leftarrow 0$
21: $\quad$ **else**
22: $\quad\quad$ Tabu $\leftarrow$ Tabu $\cup o_{new}$
23: $\quad\quad i \leftarrow i + 1$
24: $\quad$ **end if**
25: $\quad$ Update_best_result(G, o_{new})
26: **end while**
27: **return** G_{best}

Algorithm 2 EDA Baseline

Input: Population size N, selection ratio α, cost function g
Output: Best individual $\boldsymbol{x}'$ and cost found $g(\boldsymbol{x}')$

1: $G_0 \leftarrow N$ individuals randomly sampled
2: **for** $t = 1, 2, \ldots$ until stopping criterion is met **do**
3: Evaluate G_{t-1} according to $g(\cdot)$
4: $G^S_{t-1} \leftarrow$ Select top $\lfloor \alpha N \rfloor$ individuals from G_{t-1}
5: $f_{t-1}(\cdot) \leftarrow$ Learn a probabilistic model from G^S_{t-1}
6: $G_t \leftarrow$ Sample N individuals from $f_{t-1}(\cdot)$
7: **end for**

III. Estimation of Distribution Algorithms

In this section, we review the state of the art of EDAs for continuous spaces and explain the basis of our proposal.

Algorithm 2 shows the basic EDA approach for continuous spaces. Only two parameters are required, the size N of the population and the ratio of the population $\alpha \in (0, 1)$, which is selected to update the probabilistic model (line 4) according to a cost function $g(\boldsymbol{x})$ (line 3). Each generation is denoted as G_t and is sampled (line 6) from the probabilistic model $f_{t-1}(\boldsymbol{x})$ estimated with the top $\lfloor \alpha N \rfloor$ individuals from the previous generation G_{t-1} (line 5). The initial generation G_0 is sampled randomly (line 1) considering the bounds of the search space and with different sampling methods and assumptions about the density of the solutions.

Depending on the type of probabilistic model used to drive the EDA (line 5), we differentiate between the following.

1) Parametric EDAs, which assume a probability distribution for the variables involved, such as Gaussian models [10] or copula-based models [35], [36].
2) Nonparametric EDAs, which do not assume a probability distribution, but use KDEs such as KEDA [12].
3) Histogram-based models, which use histogram-based probabilistic models at each generation to describe the univariate distribution of selected individuals from the population and generate promising solutions such as histogram-based EDA [37].

Furthermore, depending on the complexity of $f_t(\boldsymbol{x})$ we distinguish between univariate, bivariate, and multivariate EDAs.

Univariate EDAs do not consider dependencies between the variables. Some examples are the continuous univariate marginal distribution algorithm (UMDA$_C$) [38] and the continuous population-based incremental learning algorithm (PBIL$_C$) [39] using independent Gaussian distributions.

Bivariate EDAs restrict each variable to depend at most on one parent variable, such as the continuous mutual information maximizing input clustering [40] which learns a chain-structured probabilistic model by adapting the concept of conditional entropy for uni and bivariate Gaussian distributions.

Multivariate EDAs do not restrict the dependencies between the variables. Some examples are the EGNA [10] and the continuous iterated density estimation algorithm [41] where a GBN is iteratively updated and sampled; the estimation of multivariate normal distribution algorithm (EMNA) [1] where a multivariate Gaussian distribution is estimated in each iteration with the best solutions of the previous iteration; and the distribution estimation using the Markov network algorithm [42], which uses Markov networks to model and sample the distribution.

In the state of the art of EDA applications for real problems where a multivariate point of view is desired, there has been a tendency to use EGNA. This is due to its competitive results compared to other algorithms, and its transparency and ease of implementation. Therefore, the possible premature convergence of EGNA has been considered in the literature. Some approaches involve modifying the variances of the variables manually so that they are not drastically reduced, such as [43] which does not allow the variance of any variable to be reduced by more than one during the runtime of the algorithm, and [44], which multiplies the variance of each variable by a constant value lower than one. The strategy of variance reduction when no better solutions are found after a certain number of iterations and increasing it when a new better solution is found to avoid falling into a local optimum was proposed in [45]. Other more complex approaches include setting a search direction based on information gathered during the execution of the algorithm. In this research line, it is worth mentioning covariance matrix adaptation (CMA-ES) [46], similar to EGNA, that is able to detect and establish a correct search direction when updating the covariance matrix and the vector of means. Other strategies include shifting the mean vector while keeping the covariance matrix [47].

However, some works [48] have determined that these approaches could be improved, if instead of using only the information of the previous generation, the information of many previous generations were also used. Having a set of solutions of the entire history visited by the EDA, similar to a tabu list is proposed. This strategy behavior similar to some differential evolution approaches such as the JADE algorithm [49], which instead of using information from good solutions, uses information from bad solutions to generate new ones. Finally, in the area of EDAs, it was proposed to update the GBN considering the best individuals of several previous generations [2], [50]. These approaches are the archive-based algorithms, where the archive length is the number of previous generations considered in the probabilistic model update.

IV. Semiparametric Estimation of Distribution Algorithm

This section describes the proposed approach. First, the main drawbacks of the traditional EGNA algorithm are analyzed, and then we introduce SPEDA. Note that EMNA has similar drawbacks to EGNA due to the Gaussianity assumption, but in this section, we will focus on EGNA since SPEDA aims to improve the results found by EGNA by using a more complex type of BN than GBNs.

A. EGNA

EGNA is one of the most extended multivariate EDAs among the state-of-the-art methods of solving optimization problems in continuous spaces. In this algorithm, a GBN is learned in each iteration using the best individuals of the

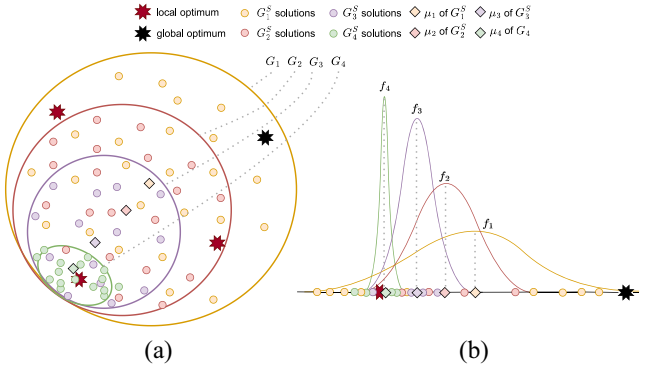


Fig. 2. Search space that is iteratively approximated by an EGNA in (a) two and (b) one dimensions, where a single global optimum (black star) and some local optima (red stars) exist. The means and the search space area defined by the solutions of each generation G_t are represented by squares and ellipses, respectively, in the 2-D landscape and by squares and a Gaussian density function in the 1-D landscape.

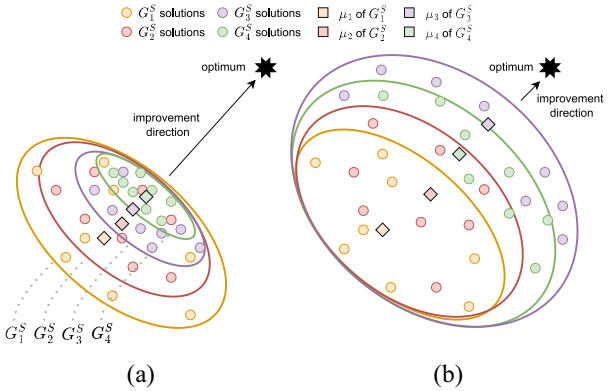


Fig. 3. Two search processes are shown in which each generation G_t is obtained by sampling a probabilistic model estimated (a) from the best individuals of the previous generation or (b) from the best individuals of the previous l generations. The means and the search area defined by the solutions of each generation G_t are represented by squares and ellipses, respectively, on the 2-D landscape.

previous iteration. This GBN is sampled to obtain the new individuals of the present generation. This loop is iterated until a stopping criterion is met following the scheme of Algorithm 2. Thus, the mean $\boldsymbol{\mu}$ at each iteration t is estimated by the mean vector

$$\boldsymbol{\mu}_t = \frac{1}{|G_{t-1}^S|} \sum_{i=1}^{|G_{t-1}^S|} \boldsymbol{x}_i^{t-1}$$

and the covariance matrix estimated by

$$\boldsymbol{\Sigma}_t = \frac{1}{|G_{t-1}^S|} \sum_{i=1}^{|G_{t-1}^S|} \left(\boldsymbol{x}_i^{t-1} - \boldsymbol{\mu}_t\right)\left(\boldsymbol{x}_i^{t-1} - \boldsymbol{\mu}_t\right)^T$$

where G_{t-1}^S is the set of solutions selected from the previous generation, $\boldsymbol{\mu}_t$ is a vector that contains the mean of each of the n variables, $\boldsymbol{x}_i^{t-1}$ denotes the ith selected individual in generation $t-1$, and $|\cdot|$ refers to the cardinality of the set.

In this work, to perform a fair comparison with the proposed algorithm, EGNA has been designed to learn the parameters of the model by maximizing the log likelihood, and to learn the structure of the GBN using the HC algorithm [30] to optimize the BIC score [31].

The EDA literature describes the search process of EGNA as a procedure in which the area of the space explored by each successive generation becomes increasingly small, until the algorithm converges to a small area in the landscape. This is represented in two dimensions in Fig. 2(a) where the space explored by each iteration and $\boldsymbol{\mu}_t$ are represented as ellipses in the landscape and different colored squares, respectively, and it is represented in one dimension in Fig. 2(b). The sizes of these ellipses are defined by the covariance matrices $\boldsymbol{\Sigma}_t$, which play a key role in the explored search space of each generation, and the search direction is influenced by the positions of the means of consecutive generations $\boldsymbol{\mu}^t$ and $\boldsymbol{\mu}^{t-1}$.

Fig. 3(a) shows how the Euclidean distance between the covariance matrix of each generation and the identity matrix becomes increasingly small as the iterations of the algorithm progress. Represented geometrically, the set of solutions selected from the previous generation G_{t-1}^S tends to lie in the semi-ellipse with a smaller distance to the global optimum being sought. By estimating a probabilistic model $f_{t-1}(\cdot)$ of this search subspace, the area defined by the new solutions G_t sampled from f_{t-1} is not only smaller than that defined by G_{t-1} but also assumes an already visited area during the algorithm runtime. This is shown in Fig. 3(a), where each space defined by G_t is a subspace of the search area represented by the solutions of G_{t-1}. This process can cause different runs of the same EGNA configuration to converge to different areas of the search space and can thus result in premature convergence or high variance in the results found between the different runs.

The high generalization power of Gaussian probability distributions, which is generally advantageous, may cause EGNA to converge to local optima, which is a disadvantage [11]. Fig. 2 shows an example in which there are some local optima and a global optimum, and the EGNA tends to converge to one of the local optima due to the location of the initially sampled solutions in the early generations.

Furthermore, restricting the dependencies between variables linearly may imply a restriction in the way the search space is explored, since it will not be possible to consider more complex relationships either in the learning of the relationships or in the sampling of new solutions.

Therefore, the use of more complex models that do not assume a Gaussian density and do not assume linear dependencies between variables may allow the exploration of subspaces of the landscape without forcing the algorithm to choose one of the two areas of the space and thus avoid falling into local optima solutions. This idea of independently exploring different areas of the landscape was previously developed by Pelikan and Goldberg [51] by restricting the search space in subareas using K-means clustering and exploring them with EDAs, as well as by Peña, Lozano, and Larrañaga [52], who used mixtures of Gaussians to model the population in each iteration. In this article, we present SPEDA as a generalization of the previous approaches where the use of mixtures is

Algorithm 3 SPEDA

Input: Population size N, selection ratio α, archive length l, cost function g

Output: Best solution $\boldsymbol{x}'$ and cost $g(\boldsymbol{x}')$ found

1: $G_0 \leftarrow N$ individuals randomly sampled
2: $i \leftarrow 0$
3: $A^t = \emptyset$
4: **for** $t = 1, 2, \ldots$ until stopping criterion is met **do**
5: Evaluate G_{t-1} according to cost function g
6: Update best solution $\boldsymbol{x}'$ obtained and compute $g(\boldsymbol{x}')$
7: $G^S_{t-1} \leftarrow$ Select top $\lfloor \alpha N \rfloor$ individuals from G_{t-1}
8: **if** $i < l$ **then**
9: $A^{t-1} \leftarrow A^{t-1} \cup G^S_{t-1}$
10: $i \leftarrow i + 1$
11: **else**
12: $A^{t-1} \leftarrow A^{t-1} \cup G^S_{t-1} \backslash G^S_{t-l-1}$
13: **end if**
14: $G_{t-1} \leftarrow$ Structure learning using HC (Algorithm 1)
15: $\boldsymbol{\Theta}_{t-1} \leftarrow$ Estimate parameters of model (Section II-C)
16: $f_{t-1}(\cdot) \leftarrow (G_{t-1}, \boldsymbol{\Theta}_{t-1})$ Build probabilistic model
17: $G_t \leftarrow$ Sample N individuals from $f_{t-1}(\cdot)$
18: **end for**

extended to the use of KDEs, but only in those variables that do not fit a Gaussian.

B. SPEDA

In this section, we present our approach to address the limitations of the traditional EGNA introduced in Section IV-A, thus improving the state-of-the-art of EDAs for continuous optimization. Algorithm 3 shows the outline of SPEDA, where SPBNs are used as a more complex probabilistic model (lines 14–16) and the new generations are sampled (line 17) considering information learned from more than one past iteration (lines 8–12).

The state-of-the-art of EDAs present different ways of initializing the algorithm, such as using a dataset as the initial generation [6] or, more commonly, initializing the algorithm from a set of solutions sampled from different possible probability distributions, such as a uniform distribution or a Gaussian distribution. In the case of SPEDA, it is not desired to bias the decisions to be made by the algorithm by defining a probability distribution over the solutions from which the algorithm is initialized. If the algorithm is initialized from a population that is randomly sampled from Gaussian distributions, successive generations of the algorithm will most likely also fit Gaussians, as will the relationships between the variables. For this reason, the algorithm is initialized from a set of uniformly sampled solutions in the search space defined by the problem to be optimized (Algorithm 3, line 1). Each variable is independently sampled from a uniform distribution, $X_i \sim \mathcal{U}(a_i, b_i)$, where a_i and b_i are the minimum and maximum bounds of X_i in the optimization landscape.

Section IV-A explains some of EGNA's limitations as a result of the Gaussianity assumption in GBNs during runtime. SPEDA overcomes this deficiency by using an SPBN. As a consequence, SPBNs allow the existence of nonlinear dependencies between variables. The key benefit is that SPEDA determines when nodes or dependencies between variables must be estimated by a KDE. If all variables and the CPDs that define the relationships found between the variables fit a Gaussian distribution, SPEDA will learn a GBN, which will make the algorithm procedure easier. Otherwise, an SPBN with some KDE variables will be learned. SPBN learning (Algorithm 3, lines 14 and 15) involves the estimation of the parameters and structure learning. For parameter estimation, the relationships between variables that are linearly Gaussian are estimated using log-likelihood maximization, while those that are CKDEs are estimated using the normal reference rule, as detailed in Section II-C. For structure learning, a modified version of the HC is executed in each iteration, which is detailed in Algorithm 1. After learning the SPBN, the joint probability distribution represented by the probabilistic model is sampled to generate new solutions (line 15). The sampling process (line 17) is implemented using the probabilistic logic sampling method [53], where the nodes are sampled in a forward direction following an ancestral order (from the parents to the children of the graph) using the evidence of the already sampled parents of the nodes.

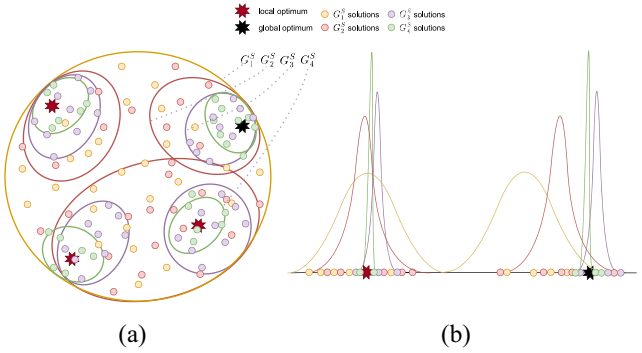


Fig. 4. Search space iteratively approximated by SPEDA in (a) two and (b) one dimensions is shown, where a single global optimum (black star) and some local optima (red stars) exist. The search space area defined by the solutions of each generation G_t are represented by ellipses and a density function on the two and 1-D landscapes, respectively.

The use of KDEs allows the parallel exploration of different areas of the search space that have a high probability density, which is unfeasible with the use of GBNs due to the assumption of (unimodal) Gaussianity. This implies that future generations may be explored and sampled in subspaces with high probability and may change during runtime. The expected behavior of both approaches are shown in Figs. 2 and 4, where a situation with several local optima is illustrated. While EGNA decides to exploit the search space area of a local optimum in Fig. 2, SPEDA is able to simultaneously explore all local optima, and converge to the global optimum in Fig. 4. Note that SPEDA in the second generation in the 2-D landscape generalizes two of the local optima into a Gaussian, but in future generations, it decides to independently explore each of the local optima.

Section III reviews some relevant studies for premature convergence in EDAs for continuous optimization. In the case of

SPEDA, we propose that the SPBN used as the main engine of the algorithm is updated by considering the best $\lfloor \alpha N \rfloor$ solutions from each of the previous l generations (Algorithm 3, line 12). Thus, in each iteration, SPEDA estimates the SPBN from A^{t-1} (lines 8–12)

$$A^{t-1} = G^S_{t-1} \cup G^S_{t-2} \cup \cdots \cup G^S_{t-l}$$

where G_t has been truncated to a size of $\lfloor \alpha N \rfloor$ by selecting the best solutions according to a cost function (line 7). Fig. 3 shows a comparison of the performance of the traditional EGNA approach (a) and the performance when considering more than the very last generation (b). Note that a search direction is established considering the best individuals of the previous l generations. With this approach, it is expected that the new solutions sampled from the learned probabilistic model $f_{t-1}(\boldsymbol{x})$ will be located in a landscape that is not in a previously explored area, but in the desired search direction determined by the information gained in earlier iterations.

Finding a balance between exploration and exploitation in the landscape is required to design an algorithm that does not converge to local optimal solutions [54]. Fig. 3 illustrates how the behavior of the standard EGNA favor an algorithm that exploits the search zones more than it explores the search space, creating an imbalance between these two characteristics. Nevertheless, the combination of SPBNs and the use of information gained from previous generations gives SPEDA a tradeoff between both traits, since KDEs allow several zones of the space to be explored simultaneously and to be found by determining a search direction during the process, allowing each zone to be exploited independently.

V. Experimental Results

In this section, we show the results of comparing our approach with some state-of-the-art optimizers in continuous environments on some well-known benchmarks, and in a real-world portfolio optimization problem. We also report on the complexity and time analysis of our approach.

Eight different algorithms are used in the experimental comparison: 1) EMNA [1]; 2) EGNA [10]; 3) SPEDA, a multivariate version of the KEDA (m_KEDA); 4) CMA-ES [46]; 5) JADE [49]; 6) SHADE [55]; and 7) L-SHADE [56]. EMNA and EGNA were chosen as continuous multivariate EDAs so that the results can be compared to those obtained with SPEDA. The pseudocode used for EGNA is fairly similar to that proposed for SPEDA where the probabilistic model, an SPBN for the case of SPEDA, and a GBN for the case of EGNA, is updated with the best solutions obtained in the previous l generations. Note that the traditional EGNA does not consider this archive-based approach, but it has been adapted to perform a fair comparison with SPEDA. The pseudocode of EMNA has been implemented traditionally, where in each iteration a multivariate Gaussian is estimated from the best individuals of the last generation. It is interesting to determine whether SPEDA significantly improves the results of EGNA and EMNA as another optimization strategy for continuous environments. The comparison also includes a multivariate version of KEDA (m_KEDA), where all the nodes and dependencies are restricted to be estimated using KDE. The algorithm shares the archive characteristic proposed for SPEDA and EGNA, so m_KEDA can be considered a particular case of SPEDA, where Gaussian variables are forbidden. The number of folds during the SPBN learning [K in (5)] for the SPEDA and m_KEDA approaches has been set to $k = 10$, as proposed in the original work [13]. Because CMA-ES is a common comparison tool for this class of algorithms, its results are also included. The JADE, SHADE, and L-SHADE algorithms were selected as members of the family of EAs in the area of differential evolution. Indeed, it has been widely used in recent years for real optimization tasks.

TABLE I
Best Parameter Configuration Found for the Algorithms After Parameter Tuning. The Parameters Include the Population Size N, the Ratio of Solutions Selected From Each Generation $\alpha \in [0, 1]$, the Archive Length l, and the Adaptation Rates of the Self-Adaptive Parameters (c) and the Greediness of the Mutation Strategy (p) of the JADE Algorithm

	EMNA	EGNA	SPEDA	m_KEDA	CMA-ES	JADE	SHADE	L-SHADE
N	300	300	300	300	4log(D)	300	300	300
α	0.4	0.6	0.4	0.4	-	-	-	-
l	-	10	15	15	-	-	-	-
c	-	-	-	-	-	0.1	-	-
p (%)	-	-	-	-	-	10	-	-

The test benchmarks are listed and characterized in the supplementary material, and were obtained from IEEE CEC2014 [57] and IEEE CEC2017 [58]. All tests are single-objective optimization problems with different difficulties grouped into unimodal–multimodal and separable–nonseparable[1] functions.

Table I shows the best parameter configuration found for the algorithms compared in this section for this set of experiments. Note that the maximum number of fitness evaluations has been set to $10\,000D$, where D is the dimension of the benchmark optimization problem. In this work, we analyze the cases of $D = 30$ and $D = 50$. Each algorithm was independently run 25 times for each benchmark and dimension. The results analyzed in this section show the mean and standard deviation of the function error value (FEV) of the 25 independent runs, where the FEV is defined as the difference between the costs of the (known) optimal solution $\boldsymbol{x}''$ and the best achieved solution $\boldsymbol{x}'$: FEV $(\boldsymbol{x}') = g(\boldsymbol{x}') - g(\boldsymbol{x}'')$. Note that a difference lower than $1e-8$ is reported as zero in the experimental results, and the objective is to minimize the FEV.

All the experiments and algorithms were implemented in Python. The code of SPEDA, m_KEDA,and benchmark implementations will be merged in the near future into the EDAspy Python package that is publicly available in a GitHub repository.[2] All the experiments and code are already available in a GitHub repository.[3] The experiments were conducted on an Ubuntu 20 machine with an Intel Core i7-6700K processor, 32 GB of RAM, and an AMD Radeon RX 460 graphic card.

[1] A function is said to be separable if it can be expressed as a mathematical operation between functions of smaller dimension.

[2] https://github.com/VicentePerezSoloviev/EDAspy

[3] https://github.com/VicentePerezSoloviev/SPEDA

TABLE II
MEAN AND STANDARD DEVIATION OF FEV AFTER 25 EXECUTIONS OF ALL BENCHMARK FUNCTIONS WITH 30 VARIABLES ($D = 30$) OBTAINED FROM EMNA, EGNA, SPEDA, M_KEDA, CMA-ES, JADE, SHADE, AND L-SHADE ALGORITHMS. THE BEST RESULT FOR EACH BENCHMARK IS HIGHLIGHTED IN BLUE

Benchmark	EMNA	EGNA	SPEDA	m_KEDA	CMA-ES	JADE	SHADE	L-SHADE
cec14_1	7.6e6 ± 7.7e5	1.5e5 ± 3.3e4	0.000 ± 0.000	2.4e4 ± 2.0e4	1.4e3 ± 1.4e3	6.6e5 ± 5.8e4	4.1e4 ± 1.1e4	0.000 ± 0.000
cec14_2	3.8e7 ± 4.5e6	1.1e7 ± 3.5e6	0.000 ± 0.000	1.7e5 ± 1.0e5	0.000 ± 0.000	1.961 ± 5.732	0.000 ± 0.000	0.000 ± 0.000
cec14_3	3.1e4 ± 1.9e3	4.741 ± 14.86	0.000 ± 0.000	0.960 ± 1.310	0.000 ± 0.000	1.813 ± 3.123	0.000 ± 0.000	2.4e3 ± 0.000
cec14_4	29.25 ± 1.232	60.48 ± 15.99	27.83 ± 0.643	28.12 ± 0.090	10.82 ± 14.28	30.38 ± 19.91	28.35 ± 32.10	6.770 ± 32.10
cec14_5	20.96 ± 0.071	20.97 ± 0.043	20.18 ± 0.024	321.2 ± 0.060	32.58 ± 0.741	20.83 ± 0.113	320.4 ± 0.060	320.6 ± 0.060
cec14_6	0.034 ± 0.012	0.011 ± 0.031	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_7	11.23 ± 12.28	0.353 ± 0.032	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_8	158.7 ± 6.490	7.862 ± 6.742	4.671 ± 1.455	225.8 ± 0.920	65.61 ± 4.991	57.07 ± 7.601	146.4 ± 16.27	36.07 ± 16.27
cec14_9	157.2 ± 9.191	126.2 ± 7.841	163.2 ± 8.161	248.3 ± 10.54	187.2 ± 4.751	186.1 ± 3.091	184.8 ± 1.940	182.7 ± 1.940
cec14_10	4.1e3 ± 360.9	4.9e3 ± 276.9	1.3e3 ± 128.6	8.8e3 ± 40.85	7.5e3 ± 484.2	6.1e3 ± 1.1e3	7.1e3 ± 758.4	3.6e3 ± 758.4
cec14_11	6.2e3 ± 846.4	5.3e3 ± 263.3	5.0e3 ± 167.2	9.2e3 ± 798.2	1.1e4 ± 697.3	9.4e3 ± 1.6e3	9.6e3 ± 1.9e3	1.7e3 ± 1.9e3
cec14_12	2.541 ± 0.232	2.521 ± 0.334	2.273 ± 0.212	3.860 ± 0.230	0.040 ± 0.031	0.071 ± 0.033	0.040 ± 0.010	0.310 ± 0.010
cec14_13	0.481 ± 0.032	0.283 ± 0.031	0.242 ± 0.010	0.530 ± 0.000	0.291 ± 0.080	0.840 ± 0.161	0.380 ± 0.070	0.190 ± 0.070
cec14_14	0.322 ± 0.011	0.672 ± 0.874	0.284 ± 0.014	0.410 ± 0.010	0.481 ± 0.184	0.694 ± 0.331	0.284 ± 0.020	0.320 ± 0.020
cec14_15	6.4e5 ± 1.5e5	9.7e9 ± 1.1e6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_16	15.00 ± 0.001	12.34 ± 0.290	12.24 ± 0.151	25.50 ± 17.68	13.68 ± 0.291	13.73 ± 0.022	12.87 ± 0.780	12.80 ± 0.780
cec14_17	2.5e4 ± 7.4e3	5.4e8 ± 5.9e8	5.470 ± 1.420	66.50 ± 75.66	181.4 ± 178.5	3.000 ± 1.540	0.010 ± 0.010	0.000 ± 0.000
cec14_18	1.5e6 ± 3.5e5	7.8e9 ± 3.8e9	14.69 ± 2.110	111.0 ± 124.4	128.4 ± 78.19	81.30 ± 1.970	1.720 ± 0.200	0.270 ± 0.200
cec14_19	211.9 ± 37.9	1.0e9 ± 8.9e8	71.43 ± 1.510	100.0 ± 1.410	69.31 ± 2.080	66.75 ± 0.950	67.47 ± 0.520	66.87 ± 0.520
cec14_20	4.5e8 ± 2.2e8	9.9e9 ± 0.000	6.560 ± 1.310	50.50 ± 70.00	85.45 ± 59.54	33.40 ± 1.560	0.460 ± 0.080	0.350 ± 0.080
cec14_21	1.4e4 ± 5.7e3	1.3e9 ± 1.1e7	1.580 ± 0.510	61.50 ± 54.45	3.790 ± 2.040	0.740 ± 0.210	0.540 ± 0.110	0.520 ± 0.110
cec14_22	2.1e7 ± 1.5e7	9.9e9 ± 0.000	24.41 ± 0.400	75.50 ± 88.39	11.42 ± 3.750	1.480 ± 0.520	0.320 ± 0.040	0.200 ± 0.040
cec14_23	1.0e4 ± 594.2	9.5e3 ± 1.260	9.5e3 ± 3.390	41.50 ± 153.4	9.5e3 ± 5.630	9.5e3 ± 5.870	9.5e3 ± 4.560	9.5e3 ± 4.560
cec14_24	6.3e3 ± 25.49	6.3e3 ± 177.8	6.1e3 ± 13.81	96.00 ± 74.95	6.1e3 ± 14.94	6.2e3 ± 39.61	6.1e3 ± 32.98	6.1e3 ± 32.98
cec14_25	5.2e3 ± 0.240	5.3e3 ± 163.7	5.2e3 ± 0.060	110.0 ± 57.98	5.2e3 ± 0.030	5.2e3 ± 0.230	5.2e3 ± 0.670	5.2e3 ± 0.670
cec14_26	1.1e4 ± 0.190	1.1e4 ± 907.9	1.1e4 ± 0.570	1.1e4 ± 13.44	1.1e4 ± 0.000	1.1e4 ± 0.320	1.1e4 ± 0.020	1.1e4 ± 0.020
cec14_27	1.1e4 ± 0.990	1.4e4 ± 1.3e3	1.1e4 ± 0.830	1.1e4 ± 28.99	1.1e4 ± 0.040	1.1e4 ± 1.700	1.1e4 ± 0.010	1.1e4 ± 0.010
cec14_28	1.2e4 ± 451.4	1.5e4 ± 8.2e2	1.1e4 ± 3.150	1.1e4 ± 64.35	1.1e4 ± 0.010	1.2e4 ± 21.63	1.1e4 ± 2.200	1.2e4 ± 2.200
cec14_29	4.5e5 ± 1.1e5	8.5e8 ± 6.5e8	7.9e3 ± 1.510	7.9e3 ± 50.20	7.9e3 ± 36.64	7.9e3 ± 1.290	7.9e3 ± 0.580	7.9e3 ± 0.580
cec14_30	3.5e7 ± 2.3e6	1.0e8 ± 0.000	8.4e3 ± 1.620	8.4e3 ± 126.5	8.4e3 ± 4.290	8.4e3 ± 7.150	8.4e3 ± 5.580	8.4e3 ± 5.580
cec17_1	3.8e7 ± 4.3e6	8.2e9 ± 3.1e9	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_2	42.19 ± 6.050	8.2e4 ± 9.6e3	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.2e4 ± 0.000
cec17_3	124.9 ± 0.260	1.2e4 ± 7.8e3	24.97 ± 0.310	101.0 ± 2.830	6.100 ± 1.830	15.72 ± 1.650	18.18 ± 0.840	15.24 ± 0.840
cec17_4	244.1 ± 10.40	8.8e4 ± 3.4e4	60.06 ± 9.250	78.01 ± 15.56	52.90 ± 11.70	184.6 ± 27.93	94.03 ± 7.660	9.910 ± 7.660
cec17_5	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	51.06 ± 67.88	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_6	1.8e3 ± 141.4	3.2e3 ± 90.44	166.2 ± 4.270	60.03 ± 12.73	51.40 ± 13.41	172.5 ± 17.55	98.23 ± 9.810	112.6 ± 9.810
cec17_7	1.2e3 ± 340.5	452.3 ± 45.42	78.90 ± 0.560	79.02 ± 28.28	91.23 ± 3.455	101.2 ± 4.455	89.45 ± 4.455	79.00 ± 0.000
cec17_8	2.840 ± 0.560	118.9 ± 3.340	0.000 ± 0.000	99.50 ± 13.44	5.100 ± 1.160	6.070 ± 1.210	7.180 ± 0.380	101.0 ± 0.380
cec17_9	7.0e3 ± 463.2	9.4e3 ± 3.0e3	6.2e3 ± 483.4	200.0 ± 141.4	1.1e3 ± 3.3e3	1.1e4 ± 4.1e3	1.1e4 ± 3.5e3	2.7e3 ± 3.5e2
cec17_10	978.7 ± 172.2	203.4 ± 203.2	5.835 ± 0.310	59.50 ± 55.86	47.33 ± 25.08	10.96 ± 1.890	10.12 ± 0.290	5.930 ± 0.490
cec17_11	5.4e6 ± 8.9e5	9.6e3 ± 5.7e3	0.000 ± 0.000	120.0 ± 141.4	3.580 ± 2.740	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_12	1.7e6 ± 4.3e5	7.7e3 ± 3.1e3	120.4 ± 5.122	139.5 ± 71.42	124.4 ± 58.86	26.84 ± 6.980	11.69 ± 1.220	4.390 ± 1.220
cec17_13	1.5e3 ± 718.5	1.3e3 ± 424.8	0.000 ± 0.000	105.5 ± 92.63	279.4 ± 196.4	10.59 ± 3.070	2.800 ± 0.720	0.000 ± 0.000
cec17_14	5.1e5 ± 1.9e5	2.2e3 ± 211.3	83.23 ± 4.556	95.59 ± 7.780	132.2 ± 167.1	13.25 ± 3.590	7.070 ± 0.230	2.370 ± 0.230
cec17_15	313.9 ± 78.40	234.1 ± 21.33	65.78 ± 34.34	84.33 ± 19.80	0.730 ± 0.190	2.030 ± 0.860	1.110 ± 1.190	0.530 ± 1.190
cec17_16	2.5e7 ± 1.2e7	1.2e3 ± 334.9	0.000 ± 0.000	77.06 ± 8.490	134.7 ± 80.60	9.680 ± 4.700	2.170 ± 0.380	0.010 ± 0.000
cec17_17	8.8e3 ± 5.2e3	982.2 ± 332.5	83.45 ± 3.450	90.51 ± 10.61	103.2 ± 65.50	1.980 ± 1.520	0.480 ± 0.020	0.500 ± 0.020
cec17_18	1.7e8 ± 5.7e7	1.2e4 ± 3.1e3	18.94 ± 2.344	20.62 ± 14.14	134.3 ± 51.24	67.84 ± 2.780	62.73 ± 0.290	60.90 ± 0.290
cec17_19	43.00 ± 5.420	23.45 ± 3.344	13.45 ± 3.344	20.71 ± 14.14	16.23 ± 4.450	4.510 ± 1.190	1.250 ± 0.180	0.250 ± 0.180

A. *Experimental Results on 30-D Benchmarks*

Table II shows the mean and standard deviation of FEV found by each algorithm after 25 independent runs for each of the benchmarks with $D = 30$, where the best results found are highlighted in blue.

On the one hand, it is important to stress the improvement found by SPEDA compared to the other EDAs versions. In almost all functions, SPEDA finds a better solution than EMNA and EGNA with a lower standard deviation. Note that EGNA improves the results found by EMNA in nearly all the functions, and that the exclusive use of CKDEs (m_KEDA) in contrast to our approach has only improved the results in the case of the composition functions (cec14_23-cec14_30).

On the other hand, the clear competitor for SPEDA in this comparison is L-SHADE, which reaches the same solutions in some of the benchmarks and outperforms SPEDA in 19 out of 49 functions. The SHADE and JADE approaches also achieve competitive results compared to SPEDA, improving those found by SPEDA in 14 and 11 functions, respectively. CMA-ES and m_KEDA algorithms beat our approach in 5 and 9 benchmarks, respectively. Additionally, SPEDA achieves the lowest FEV standard deviation in most of the experiments.

In terms of the type of optimized functions, SPEDA is able to find the optimum in all runs for unimodal functions (cec14_1, cec14_2, cec14_3, cec17_1, cec17_2), while its competitors are not able to do so. For the cec14_4 benchmark, since there is a very narrow valley between the local and global optima, SPEDA seems not to have sufficiently exploited the search space, thus preventing it from beating CMA-ES and L-SHADE. It is worth mentioning the pair of benchmarks cec14_8 and cec14_9, as they are the same function, with the difference that one is separable (cec14_8) and the other is not (cec14_9). In this case, using a GBN instead of an SPBN improves the results for the nonseparable function. The pair cec14_10 and cec14_11 has the same feature, and for the nonseparable function (cec14_11), a similar result

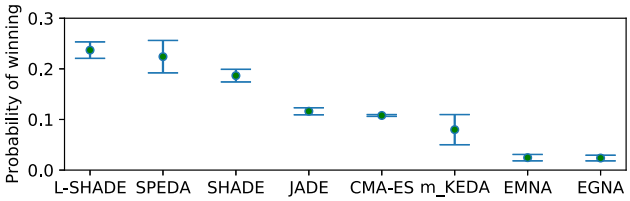


Fig. 5. Credible intervals (5% and 95% quantiles) and expected probability of winning (green dots) for results shown in Table II.

is obtained for both EGNA and SPEDA. The standard deviation for these two problems is high for all algorithms, but again the best result is that of SPEDA. However, in both pairs of benchmarks (cec14_8-cec14_9 and cec14_10-cec14_11), SPEDA improves the other algorithms for separable functions (cec14_8 and cec14_10). The cec14_12 benchmark is also nonseparable, and the algorithm that provides the best results is SHADE. As for hybrid functions (cec14_17 to cec14_22, and cec17_10 to cec17_19), the best results are found by SPEDA and L-SHADE approaches. In the case of composition functions (cec14_23 to cec14_30), which are multimodal, SPEDA, CMA-ES, and SHADE achieve the best results in most of the benchmarks, only beaten by m_KEDA. Finally, regarding the optimization of functions where the number of local optima is large (cec14_8 to cec14_11, cec17_3 to cec17_5 and cec17_7 to cec17_9), SPEDA is the best performing algorithm. Indeed, it is able to find the best solutions in 5 out of the 10 functions with this feature, followed by L-SHADE (3 out of 10) and CMA-ES (2 out of 10). This suggests that SPEDA is able to avoid local optima better than its competitors. Thus, SPEDA and L-SHADE seem to ensure better results in most of the benchmarks compared to the rest of the algorithms, finding the best results both in 26 out of the 49 benchmarks.

To conclude this comparison for the 30-D experiments, in the case of the family of EDAs, using a more complex probabilistic model, such as SPBN improves the results compared to those of the multivariate Gaussian and the GBNs. SPEDA is a competitive approximation compared to other state-of-the-art EDAs, CMA-ES, and differential evolution approaches. Moreover, it is able to provide results with a low variance after different independent executions. In addition, the experiments have shown that SPEDA always improves the results for at least unimodal and separable functions.

The results shown in Table II have been statistically analyzed. Fig. 5 shows the credibility intervals (5% and 95%) and expected probabilities of each algorithm being the winner under the posterior distribution calculated in the Bayesian analysis using the Plackett–Luce model [59]. The plot shows that L-SHADE and SPEDA are the most probable winners where the former shows a lower associated uncertainty. EMNA and EGNA are the less likely approaches to be the winners achieving a similar expected probability and associated uncertainty; thus we conjecture that both approaches perform similar. m_KEDA is the approach with higher uncertainty, but with less chances to be the winner. The plot corroborates the results shown in Table II where JADE and CMA-ES achieve a very similar performance, being the latter the one with less uncertainty. SHADE chases SPEDA in the ranking of expected probabilities to be the winner approach.

Analyzing the behavior of SPEDA during runtime, it is observed that the initialization of the algorithm has a significant impact on the predominant probabilistic model used during runtime. When starting from Gaussian distributions, SPEDA behaves in a way that favors Gaussians over CKDEs from the outset. However, when starting from a uniform probability distribution, as detailed in Section IV-B, SPEDA chooses the probabilistic model to use. The number of nodes that are fitted by CKDE increases as the algorithm's evolve, as shown in Fig. 6, where the mean and standard deviation of the percentage of CKDE-estimated nodes in each iteration are shown for the first ten benchmarks. In most cases, SPEDA tends to adjust all the nodes by CKDE. Note that in the iterations where 0% of the nodes are estimated by CKDE, the learned probabilistic model is a GBN, where the optimization process is the same as in EGNA but conditioned to the previous iterations, where an SPBN was learned. Fig. 6 shows that atypically, for the cec14_3 and cec14_9 benchmarks, nearly all nodes are Gaussian from the beginning of the algorithm execution, and this directly affects the results found. In cec14_3, the fact that an SPBN rather than a GBN was used in the first iterations of SPEDA suggests that SPEDA and EGNA are positioned in different areas of the search space, with the area explored by SPEDA being the area of the global optimum. A similar situation occurred for cec14_9, in this case in favor of EGNA. Similar results were obtained for the rest of the benchmarks, but were not included in Fig. 6 for aesthetic reasons. In the experiments we have seen a tendency for all functions to converge to 100% of CKDE-estimated nodes. This can be explained because individuals sampled from Gaussians and then selected according to the objective function are more likely to be fitted by a KDE than vice versa.

For deeper insights in SPEDA performance, we aim to analyze the potential ability of the approach to avoid the local optima in the landscape. If the variance of the solutions in the same iteration of the algorithm tends to decrease over the runtime, then it may suggest that the algorithm is converging to a smaller area, which is probably contained in the area of the previous generation. However, if the variance increases and decreases over the runtime, it can be said that the algorithm is seeking a balance between exploration and exploitation of the search space, and most likely will be able to avoid local optima. This reduction of variance between individuals is related to genetic drift [60], which is present in EDAs.

Fig. 7 shows a univariate analysis of the variance reduction of the best solutions along the runtime for the EGNA, SPEDA, and m_KEDA approaches, for the cec14_4 benchmark ($D = 30$). It is observed that EGNA early reduces the variance to zero, leading to a nearly monotonic decreasing shape. This might reveal convergence to a local optima solution. Nevertheless, SPEDA and m_KEDA suggest a decreasing tendency, but much slower than EGNA. Several ups and downs during the runtime can be appreciated, which might suggest that our approach is able to avoid local optima by using CKDE nodes (Fig. 6) and exploring more than one area at the same time. Table II shows that both approaches converge to better solutions than EGNA approach for this benchmark. Note that, although Fig. 7 shows the performance of a single variable, the rest of features have similar performance (not shown).

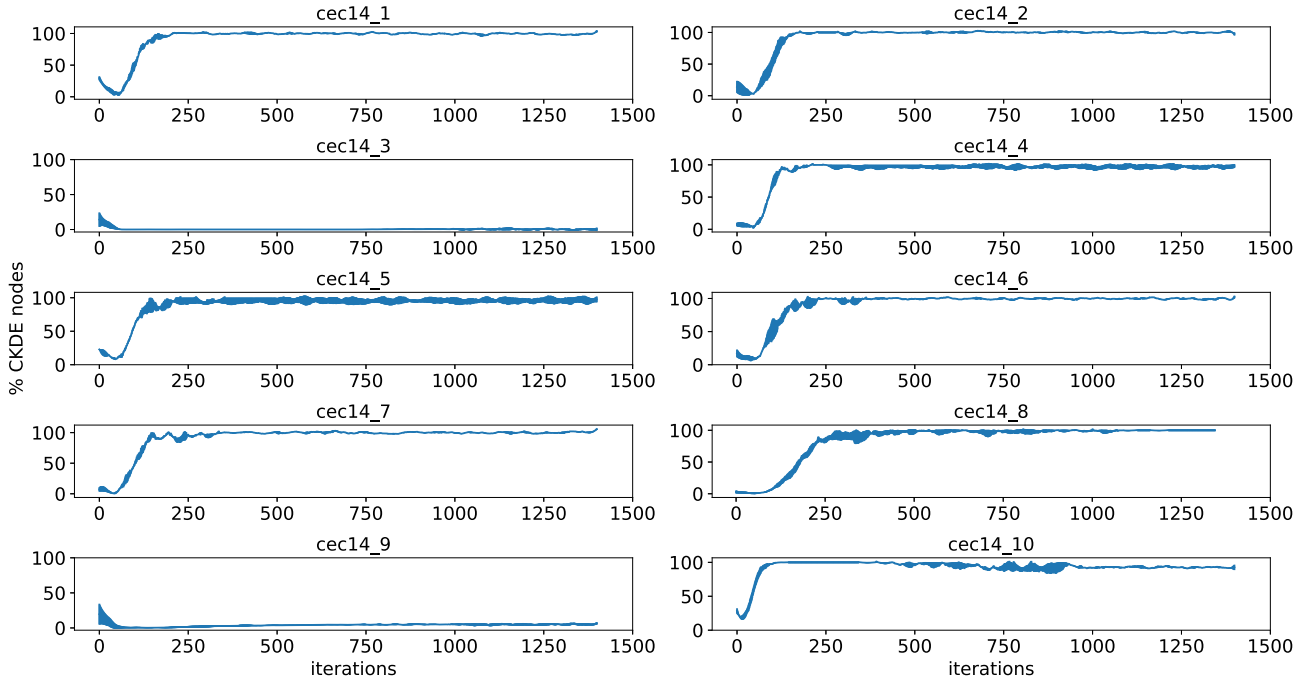


Fig. 6. Percentage of CKDE nodes during runtime, mean, and standard deviation of 25 independent executions. The experimental results for the first ten benchmarks are shown with $D = 30$.

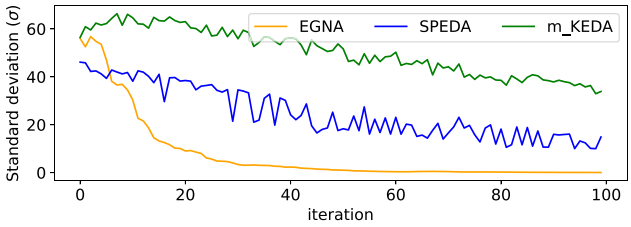


Fig. 7. Mean standard deviation (y-axis) between the best solutions in the same iteration during runtime (x-axis) after executing EGNA, SPEDA, and m_KEDA approaches 25 independent times, for the cec14_4 benchmark ($D = 30$). Note that this plot represents one variable out of D.

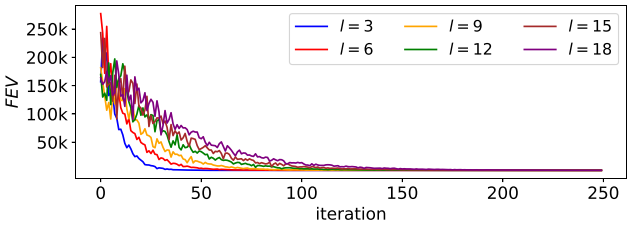


Fig. 8. Mean best cost found FEV by SPEDA for the cec14_3 for different archive lengths (l) during runtime after executing each experiment 25 independent times.

Fig. 8 analysis the convergence speed compared to the archive length (l) for the cec14_3 benchmark. The higher the l value, the slower the speed of convergence. Although for this case SPEDA reaches the global optimum regardless of l, after the hyperparameter tuning we have decided to use $l = 15$ as the general optimum parameter for all benchmarks in CEC2014 and CEC2017 for both $D = 30$ and $D = 50$.

B. Experimental Results on 50-D Benchmarks

Table III shows the mean and standard deviation of the FEV of each algorithm after 25 independent runs for each of the benchmarks with $D = 50$, where the best results for each function are highlighted in blue.

As in the 30-D case, SPEDA improves the results found by EMNA, EGNA, and m_KEDA, although m_KEDA provides competitive results compared to our approach. Although it is not as remarkable as in the case of $D = 30$, SPEDA provides the lowest variance among the results found in almost all benchmarks, since the use of the CKDEs reduces it. This is the case in m_KEDA, which also shows a low standard deviation, compared to the other EDAs. The results show that SPEDA is able to converge to the best solutions in 30 out of the 49 functions. The main competitor for the SPEDA approach in this comparison is L-SHADE, reaching the best solution in 22 out of the 49 functions. Moreover, L-SHADE outperforms the results found by SPEDA in 17 functions and SPEDA outperforms L-SHADE in 18 functions.

SPEDA achieves the best results for all the unimodal functions except for the cec14_2, in which CMA-ES, SHADE, and L-SHADE win it. Regarding the separable functions (cec14_8 and cec14_10), SPEDA converges to the best solution in all runs for both cases. Regarding the hybrid functions, L-SHADE approach is the best in this characteristic, followed by our proposal. SPEDA is the only approach able to find the best results in composition functions, as in the 30 dimensions case. Note that, in the case of $D = 30$, m_KEDA achieved good results for these functions, while no such results are shown in the case of $D = 50$. This may suggest that the combination of Gaussian and CKDE nodes scales better from the optimization point of view, compared to the exclusive use of CKDEs. Finally,

TABLE III
MEAN AND STANDARD DEVIATION OF FEV AFTER 25 EXECUTIONS ON ALL BENCHMARK FUNCTIONS WITH 50 VARIABLES ($D = 50$) OBTAINED FROM THE EMNA, EGNA, SPEDA, M_KEDA, CMA-ES, JADE, SHADE, AND L-SHADE ALGORITHMS. THE BEST RESULT FOR EACH BENCHMARK IS HIGHLIGHTED IN BLUE

Benchmark	EMNA	EGNA	SPEDA	m_KEDA	CMA-ES	JADE	SHADE	L-SHADE
cec14_1	1.1e7 ± 1.0e6	2.7e7 ± 3.e06	6.2e3 ± 1.5e4	3.6e9 ± 1.5e8	4.1e5 ± 1.4e5	6.2e6 ± 2.7e6	5.1e6 ± 1.9e6	4.8e8 ± 1.9e6
cec14_2	1.1e7 ± 3.1e6	2.3e3 ± 1.1e3	7.000 ± 14.78	2.8e7 ± 2.8e6	0.000 ± 0.000	1.6e4 ± 1.2e4	0.000 ± 0.000	0.000 ± 0.000
cec14_3	3.9e4 ± 1.5e3	0.440 ± 1.670	0.000 ± 0.000	3.5e5 ± 2.3e5	1.1e3 ± 558.6	8.2e3 ± 4.5e3	477.7 ± 409.1	3.8e4 ± 409.1
cec14_4	210.7 ± 3.340	171.2 ± 14.23	151.2 ± 15.08	76.16 ± 4.000	43.40 ± 17.65	77.91 ± 38.00	158.3 ± 54.14	96.32 ± 54.14
cec14_5	21.16 ± 0.030	21.17 ± 0.030	21.13 ± 0.060	321.2 ± 0.040	32.15 ± 0.640	21.01 ± 0.120	320.9 ± 0.050	320.8 ± 0.050
cec14_6	0.030 ± 0.000	0.010 ± 0.010	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_7	0.960 ± 0.090	2.100 ± 5.330	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_8	114.5 ± 9.080	339.6 ± 11.82	180.6 ± 10.52	473.5 ± 10.33	302.1 ± 7.802	188.3 ± 13.52	260.3 ± 13.03	195.9 ± 13.03
cec14_9	335.3 ± 17.65	338.6 ± 11.03	247.5 ± 16.05	494.9 ± 29.73	425.8 ± 11.14	416.1 ± 21.13	388.3 ± 54.55	88.18 ± 54.55
cec14_10	8.6e3 ± 911.3	1.0e4 ± 410.0	6.1e3 ± 288.1	1.7e4 ± 468.9	1.8e4 ± 2.3e3	1.1e4 ± 4.1e3	1.5e4 ± 3.6e3	1.8e4 ± 3.6e3
cec14_11	1.7e4 ± 782.1	1.0e4 ± 375.1	1.2e4 ± 73.73	1.6e4 ± 469.5	5.9e3 ± 847.0	1.1e4 ± 1.8e3	8.8e3 ± 1.2e3	8.3e3 ± 1.2e3
cec14_12	3.610 ± 0.290	3.460 ± 0.260	3.430 ± 0.590	4.470 ± 0.570	0.020 ± 0.010	0.130 ± 0.080	0.020 ± 0.020	0.530 ± 0.020
cec14_13	0.780 ± 0.020	0.520 ± 0.050	0.400 ± 0.020	0.770 ± 0.070	0.600 ± 0.090	1.080 ± 0.120	0.680 ± 0.090	0.580 ± 0.090
cec14_14	0.720 ± 0.040	0.540 ± 0.110	0.440 ± 0.020	0.920 ± 0.250	0.620 ± 0.280	0.990 ± 0.330	0.450 ± 0.140	0.480 ± 0.140
cec14_15	2.3e6 ± 4.6e5	9.3e7 ± 1.4e6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_16	25.00 ± 0.010	22.33 ± 0.670	22.23 ± 0.260	23.59 ± 0.030	22.46 ± 0.460	22.36 ± 0.020	24.45 ± 1.231	23.23 ± 0.012
cec14_17	1.4e5 ± 3.4e4	960.5 ± 80.99	34.90 ± 3.320	5.6e3 ± 191.9	181.4 ± 178.5	10.15 ± 3.680	15.28 ± 2.270	0.000 ± 0.000
cec14_18	7.4e6 ± 8.3e5	810.4 ± 40.94	68.72 ± 6.880	5.0e3 ± 2.5e3	128.4 ± 78.19	12.83 ± 6.440	35.79 ± 3.620	1.130 ± 3.620
cec14_19	1.1e3 ± 217.5	337.7 ± 80.03	209.2 ± 0.820	263.4 ± 69.36	69.31 ± 2.080	192.2 ± 1.890	198.8 ± 0.710	194.8 ± 0.710
cec14_20	195.0 ± 20.31	99.50 ± 0.000	34.97 ± 5.800	64.32 ± 1.345	85.45 ± 59.54	6.220 ± 2.990	14.08 ± 1.560	0.480 ± 1.560
cec14_21	8.5e4 ± 1.6e4	200.3 ± 12.34	8.150 ± 0.600	12.10 ± 0.812	3.790 ± 2.040	7.280 ± 0.900	4.690 ± 0.770	1.840 ± 0.770
cec14_22	6.2e8 ± 4.3e8	1.1e3 ± 12.12	25.62 ± 0.200	17.32 ± 0.412	11.42 ± 3.750	2.750 ± 0.750	1.690 ± 0.130	0.430 ± 0.130
cec14_23	9.5e3 ± 1.141	1.0e4 ± 525.2	9.4e3 ± 0.130	1.1e4 ± 683.4	9.5e3 ± 5.630	9.6e3 ± 10.65	9.5e3 ± 6.240	9.6e3 ± 6.240
cec14_24	6.7e3 ± 31.31	6.6e3 ± 238.9	6.6e3 ± 0.520	6.6e3 ± 2.410	6.6e3 ± 14.94	6.6e3 ± 27.86	6.6e3 ± 65.65	6.6e3 ± 65.65
cec14_25	5.2e3 ± 1.040	5.2e3 ± 72.15	5.2e3 ± 0.090	5.7e3 ± 41.11	5.2e3 ± 0.030	5.2e3 ± 1.600	5.2e3 ± 1.340	5.2e3 ± 1.340
cec14_26	1.1e4 ± 0.101	1.1e4 ± 0.211	1.0e4 ± 0.090	1.1e4 ± 51.29	1.0e4 ± 0.000	1.104 ± 1.220	1.0e4 ± 0.430	1.1e4 ± 0.430
cec14_27	1.1e4 ± 1.170	1.1e4 ± 4.801	1.1e4 ± 0.190	2.1e4 ± 885.4	1.1e4 ± 0.040	1.1e4 ± 11.61	1.1e4 ± 3.600	1.1e4 ± 3.840
cec14_28	1.1e4 ± 35.14	1.2e4 ± 35.14	1.1e4 ± 10.77	1.2e3 ± 182.7	1.2e4 ± 0.010	1.1e4 ± 24.67	1.2e4 ± 9.440	1.3e4 ± 9.440
cec14_29	1.7e6 ± 2.3e5	6.9e4 ± 1.2e3	7.9e3 ± 1.670	8.1e3 ± 0.400	7.9e3 ± 36.64	7.9e3 ± 2.670	7.9e3 ± 0.470	7.9e3 ± 0.000
cec14_30	5.0e8 ± 3.0e6	5.6e4 ± 237.1	8.4e3 ± 139.9	8.4e3 ± 79.91	8.4e3 ± 4.290	8.4e3 ± 2.540	8.4e3 ± 0.100	8.4e3 ± 0.100
cec17_1	9.8e7 ± 7.2e5	1.0e6 ± 2.1e5	0.000 ± 0.000	2.9e7 ± 4.7e4	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_2	213.98 ± 3.51	9.5e4 ± 3.0e4	0.000 ± 0.000	3.1e5 ± 6.5e3	0.000 ± 0.000	302.6 ± 256.1	0.000 ± 0.000	101.1 ± 0.010
cec17_3	156.57 ± 0.74	8.3e4 ± 9.4e4	120.4 ± 7.860	85.40 ± 2.640	6.040 ± 1.830	92.10 ± 34.02	107.3 ± 39.84	2.5e3 ± 39.84
cec17_4	480.6 ± 2.620	8.5e4 ± 1.6e4	348.2 ± 1.250	496.5 ± 14.93	52.93 ± 11.69	389.1 ± 32.90	334.0 ± 6.920	89.80 ± 6.920
cec17_5	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_6	5.4e7 ± 1.2e3	3.6e6 ± 1.6e4	329.9 ± 22.70	1.5e3 ± 188.1	51.40 ± 13.41	423.3 ± 29.72	325.7 ± 21.03	103.6 ± 21.03
cec17_7	136.7 ± 20.54	80.10 ± 5.330	77.92 ± 1.233	81.21 ± 3.212	101.1 ± 21.11	93.34 ± 12.12	82.23 ± 22.22	79.99 ± 1.211
cec17_8	114.5 ± 0.950	145.3 ± 1.150	0.000 ± 0.000	8.840 ± 9.650	5.130 ± 1.160	27.30 ± 2.660	25.17 ± 2.590	0.200 ± 2.590
cec17_9	1.3e4 ± 823.3	1.3e4 ± 364.3	1.1e4 ± 259.2	1.6e4 ± 348.2	1.1e4 ± 3.4e3	1.1e4 ± 1.8e3	9.2e4 ± 236.3	7.6e4 ± 236.3
cec17_10	3.3e3 ± 606.5	9.5e4 ± 360.0	91.73 ± 9.210	7.1e6 ± 1.2e3	47.33 ± 25.08	17.62 ± 2.780	44.52 ± 3.960	15.74 ± 3.960
cec17_11	1.5e7 ± 2.5e6	1.1e8 ± 1.1e3	0.000 ± 0.000	2.6e3 ± 400.6	3.580 ± 2.740	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_12	8.5e7 ± 5.5e6	1.1e5 ± 3.1e4	92.48 ± 2.160	320.1 ± 20.23	124.4 ± 58.86	124.4 ± 8.920	95.58 ± 7.080	101.6 ± 7.080
cec17_13	1.3e4 ± 2.3e3	4.3e8 ± 5.8e7	0.000 ± 0.000	0.000 ± 0.000	279.4 ± 196.4	34.77 ± 8.660	34.00 ± 0.730	0.000 ± 0.000
cec17_14	2.6e6 ± 2.8e5	3.2e9 ± 5.5e6	4.552 ± 0.122	181.3 ± 21.11	132.2 ± 167.2	21.46 ± 2.780	30.74 ± 2.890	10.67 ± 2.890
cec17_15	1.3e3 ± 201.4	1.4e4 ± 760.1	8.348 ± 1.222	12.12 ± 1.223	0.730 ± 0.190	11.47 ± 0.920	9.070 ± 0.320	6.550 ± 0.320
cec17_16	9.2e8 ± 8.4e7	9.9e5 ± 0.000	15.67 ± 4.455	83.33 ± 3.222	134.7 ± 80.60	21.95 ± 6.650	25.44 ± 2.340	1.530 ± 2.340
cec17_17	4.1e4 ± 1.5e4	5.9e5 ± 7.4e3	12.34 ± 5.334	17.56 ± 2.112	103.2 ± 65.50	15.71 ± 5.900	7.180 ± 0.260	0.510 ± 0.260
cec17_18	9.1e8 ± 1.8e8	9.9e6 ± 0.000	20.36 ± 0.944	123.1 ± 34.33	34.30 ± 51.24	87.02 ± 3.940	101.7 ± 0.430	81.03 ± 0.430
cec17_19	76.48 ± 6.470	4.2e3 ± 943.5	13.87 ± 2.153	54.45 ± 0.122	16.23 ± 4.450	12.35 ± 3.230	11.52 ± 0.460	1.270 ± 0.460

regarding the optimization of functions where the number of local optima is large, SPEDA is the best performing algorithm (6 out of 10), as in the 30 dimensions case, followed by CMA-ES (5 out of 10).

The results shown in Table III have been statistically analyzed. Fig. 9 shows the credibility intervals (5% and 95%) and expected probabilities of each algorithm being the winner under the posterior distribution calculated in the Bayesian analysis using the Plackett–Luce model [59]. It is shown that SPEDA is the best approach, where its lower bound is higher than the upper one of its competitors. SPEDA is followed by L-SHADE and SHADE, where the latter is the one with highest uncertainty. JADE and CMA-ES imitate the results analyzed for 30 dimensions. The approaches that have the least chances of being the winners are again m_KEDA, EMNA, and EGNA, where in this analysis m_KEDA achieves very low uncertainty.

Based on the results of the experiments for $D = 30$ and $D = 50$, we conclude that SPEDA can be a competitive tool for continuous optimization compared to some state-of-the-art population-based approaches. Indeed, it is able to converge to solutions with low variance in independent algorithm executions. Furthermore, the optimal landscapes for our approach seem to be functions that are unimodal separable, although it still outperforms its competitors in most of the nonseparable and multimodal functions. A good performance has also been identified in the composition functions and landscapes with a large number of local optima, regardless of the dimension.

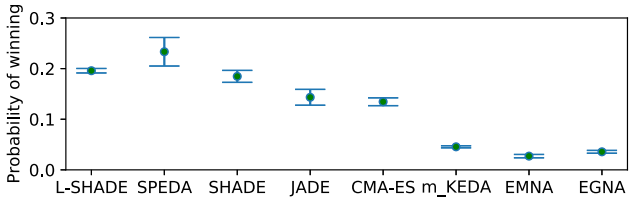


Fig. 9. Credible intervals (5% and 95% quantiles) and expected probability of winning (green dots) for results shown in Table III.

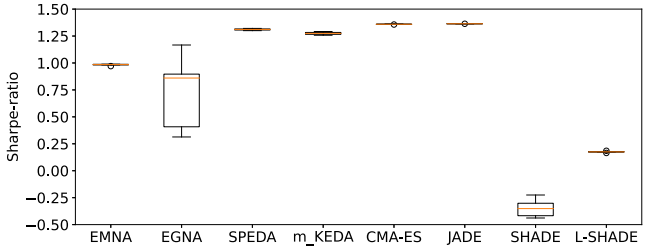


Fig. 10. Boxplot with the best results found by each algorithm after 25 independent executions for the specific portfolio optimization problem.

C. Portfolio Optimization

In this section, we compare the previously tested algorithms in a real-world portfolio optimization problem.

The portfolio optimization problem is based on the diversification aspect of the investment, where investors diversify their investments into different types of assets. The objective of this optimization task is to maximize the return of the investment but also to minimize its risk. This biobjective problem is reduced to a single-objective task by using the Sharpe-ratio metric [61], combining both aspects as follows:

$$\text{Sharpe\_ratio} = \frac{R_p - R_f}{\sigma_p} \tag{7}$$

where R_p, R_f, and σ_p are the return of the portfolio, the risk of the investment and the standard deviation of the portfolio's excess return for a series of time intervals, respectively.

The cost functions computations were made using PyPortfolioOpt Python package [62], including historical daily stock prices of 20 different assets ($D = 20$) from 29 December 1989 to 11 April 2018, also available in PyPortfolioOpt.[4]

Fig. 10 shows the Sharpe-ratio boxplot of the best solutions achieved for different algorithms. The maximum number of iterations has been limited to 300. It is observed that SPEDA, CMA-ES, and JADE are the approaches that achieve the best solutions in terms of Sharpe-ratio maximization compared to its competitors. Good results are also found by m_KEDA. In this case, the results found by EMNA improve those found by EGNA, which have a large dispersion between the solutions. Note that the median of the EGNA solutions gives us a hint about the presence of extreme data in the lower bound of the boxplot. CMA-ES and JADE approaches seem to converge to a unique solution in all the executions. While the SHADE and L-SHADE approaches offer good solutions for the benchmarks studied, they do perform well on this real problem. However, our approach maintains good performance on both types of optimization problems. The results shown in Fig. 10 have been analyzed using statistical tests to reject the null hypothesis of equal means between the different methods, and obtained a p-value of 1.75e-16. Thus, a statistically significant difference was found between the performance of the optimizers. However, analyzing the statistical tests by pairs, there is no statistically significant difference between the results obtained by EMNA and EGNA.

[4]https://raw.githubusercontent.com/robertmartin8/PyPortfolioOpt/master/tests/resources/stock_prices.csv

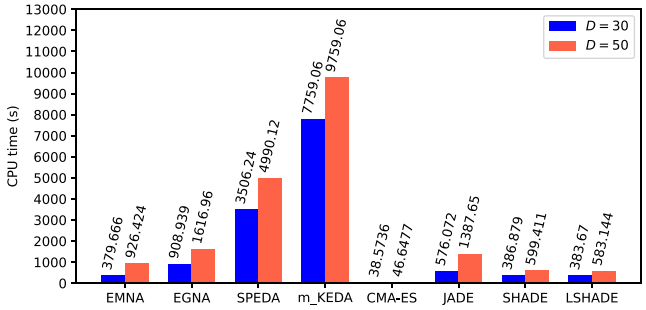


Fig. 11. Comparison of the average CPU time (in seconds) after 25 independent executions on each of the benchmarks in CEC2014 and CEC2017, for EMNA, EGNA, m_KEDA, SPEDA, CMA-ES, JADE, SHADE, and LSHADE. The results for 30 and 50 dimensions are shown in blue and red, respectively.

D. CPU Time and Complexity Analysis

In this section, we analyze the average CPU time spent during the execution of the algorithms and the asymptotic time complexity of SPEDA.

Fig. 11 shows a CPU time comparison between all the approaches used for the benchmarks. It is observed that m_KEDA is the most expensive and CMA-ES the fastest. Analyzing the four EDA, the higher the complexity of the algorithm is, the longer the execution time. Thus, m_KEDA has a longer execution time than SPEDA, which in turn takes longer than EGNA, which takes longer than EMNA. The three differential evolution variants have a similar average CPU time to that found for EMNA, where JADE is the most expensive.

The increase in SPEDA complexity compared to that of EGNA is caused by the cross-validated cost function that evaluates CKDE and Gaussian nodes, as discussed in Section II-C3. Considering this, the complexity of SPEDA is simplified as $\mathcal{O}(tOi\lambda KT)$, where t is the number of iterations of SPEDA. Fig. 11 shows that, in general, m_KEDA is more complex than SPEDA due to the cross-validated function over all the CKDE nodes, which is D in each iteration, in contrast to SPEDA, which is D at most. The cross-validated function used by SPEDA, increases the CPU time but also yields the best results, as shown in Tables II and III. Note that tuning the hyperparameter K (number of folds) might reduce the computation time but also may lead to poorest solutions.

VI. Conclusion

Traditional EDAs typically use Gaussian distributions to optimize continuous functions such as EGNA or EMNA, which use GBNs and multivariate Gaussian probability distributions, respectively. Nevertheless, using these types of probabilistic models implies assuming Gaussian distributions that only consider linear relationships between variables. In this work, we propose semiparametric EDAs, which learn a SPBN at each iteration, with the coexistence of nodes that are fitted by CKDE and nodes that assume Gaussianity. SPEDA decides iteratively whether Gaussians or CKDEs are fitter at each node.

Moreover, the traditional EGNA usually learns a GBN in each iteration by considering only the best solutions of the last iteration, which may lead to a high variance between the

solutions in independent runs of the algorithm, or convergence to local optimal solutions. SPEDA has been designed to overcome this limitation by learning a probabilistic model in each iteration considering the best solutions of l previous iterations. This is intended to establish a search direction in the landscape based on the learned information.

The empirical results showed a comparison of SPEDA with some of the most widely used EDAs for continuous function optimization, like EMNA and EGNA. We conclude that using a more complex probabilistic model, such as an SPBN, improves the results compared to those of EGNA and EMNA. SPEDA was also compared to CMA-ES, which is a similar case to EGNA; to JADE, SHADE, and L-SHADE as members of differential evolution algorithms family; and to the multivariate KDE EDA, the extreme case of SPEDA where the Gaussian nodes are forbidden. The experiments were run on 49 benchmarks in 30- and 50-D spaces, and it was found that SPEDA provided the best results in most of the benchmarks that considered landscapes of different characteristics. The results found were analyzed using a Bayesian performance analysis with the Plackett–Luce model to estimate the probability of each algorithm to be the winner approach. L-SHADE was the most probable algorithm for the case of 30 dimensions followed by SPEDA with similar probabilities. A similar performance was observed for the case of 50 dimensions where SPEDA is the most probable approach. The experimental results also include a real world optimization problem, where statistical significant differences were found, being SPEDA one of the best performing approaches, together with CMA-ES and JADE.

SPEDA is a tool that can be of great benefit for optimization tasks in complex continuous environments where the variance between the solutions proposed by our approach in different executions is low, without sacrificing the quality of the solutions. The experiments showed that learning SPBNs in each iteration can be slower than learning GBNs, so future research would include shortening this learning time or reducing the number of times the structure is learned during runtime.

ACKNOWLEDGMENT

The authors would like to thank Schloss Dagstuhl–Leibniz-Zentrum für Informatik for facilitating the Dagstuhl Seminar 22182 (https://www.dagstuhl.de/22182) entitled *Estimation of Distribution Algorithms: Theory and Applications*, which brought about several discussions on this topic.

REFERENCES

[1] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. New York, NY, USA: Springer, 2001.

[2] Y. Liang, Z. Ren, X. Yao, Z. Feng, A. Chen, and W. Guo, "Enhancing Gaussian estimation of distribution algorithm by exploiting evolution direction with archive," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 140–152, Jan. 2020.

[3] W. Dong, T. Chen, P. Tiňo, and X. Yao, "Scaling up estimation of distribution algorithms for continuous optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 6, pp. 797–822, Dec. 2013.

[4] A. Shirazi, J. Ceberio, and J. A. Lozano, "EDA++: Estimation of distribution algorithms with feasibility conserving mechanisms for constrained continuous optimization," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 1144–1156, Oct. 2022.

[5] D. Dasgupta and Z. Michalewicz, *Evolutionary Algorithms in Engineering Applications*. Heidelberg, Germany: Springer, 2014.

[6] V. P. Soloviev, P. Larrañaga, and C. Bielza, "Estimation of distribution algorithms using Gaussian Bayesian networks to solve industrial optimization problems constrained by environment variables," *J. Combinatorial Optim.*, vol. 44, pp. 1077–1098, Jul. 2022.

[7] A. Slowik and H. Kwasnicka, "Evolutionary algorithms and their applications to engineering problems," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12363–12379, 2020.

[8] J. Ceberio, A. Mendiburu, and J. A. Lozano, "A roadmap for solving optimization problems with estimation of distribution algorithms," *Nat. Comput.*, vol. 2022, pp. 1–15, Sep. 2022.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control and Artificial Intelligence*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[10] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization in continuous domains by learning and simulation of Gaussian networks," in *Proc. Genet. Evol. Comput. Congr.*, 2000, pp. 201–204.

[11] G. Neumann and D. Cairns, "Introducing intervention targeting into estimation of distribution algorithms," in *Proc. 27th Annu. ACM Symp. Appl. Comput.*, 2012, pp. 220–225.

[12] N. Luo and F. Qian, "Evolutionary algorithm using kernel density estimation model in continuous domain," in *Proc. 7th Asian Control Conf.*, 2009, pp. 1526–1531.

[13] D. Atienza, C. Bielza, and P. Larrañaga, "Semiparametric Bayesian networks," *Inf. Sci.*, vol. 584, pp. 564–582, Jan. 2022.

[14] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.

[15] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[16] C. Bielza and P. Larrañaga, "Bayesian networks in neuroscience: A survey," *Front. Comput. Neurosci.*, vol. 8, p. 131, Oct. 2014.

[17] C. Puerto-Santana, P. Larrañaga, and C. Bielza, "Autoregressive asymmetric linear Gaussian hidden Markov models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4642–4658, Sep. 2022.

[18] B. Thiesson, C. Meek, D. M. Chickering, and D. Heckerman, "Learning mixtures of DAG models," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, 1998, pp. 504–513.

[19] S. Dasgupta, "Learning mixtures of Gaussians," in *Proc. 40th Annu. Symp. Found. Comput. Sci.*, 1999, pp. 634–644.

[20] S. Moral, R. Rumí, and A. Salmerón, "Mixtures of truncated exponentials in hybrid Bayesian networks," in *Proc. Eur. Conf. Symbolic Quantitative Approaches Reason. Uncertainty*, 2001, pp. 156–167.

[21] P. P. Shenoy and J. C. West, "Inference in hybrid Bayesian networks using mixtures of polynomials," *Int. J. Approx. Reason.*, vol. 52, no. 5, pp. 641–657, 2011.

[22] H. Langseth, T. D. Nielsen, R. Rumı, and A. Salmerón, "Mixtures of truncated basis functions," *Int. J. Approx. Reason.*, vol. 53, no. 2, pp. 212–227, 2012.

[23] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Boca Raton, FL, USA: Routledge, 2018.

[24] A. Pérez, P. Larrañaga, and I. Inza, "Bayesian classifiers based on kernel density estimation: Flexible classifiers," *Int. J. Approx. Reason.*, vol. 50, no. 2, pp. 341–362, 2009.

[25] S.-C. Wang, R. Gao, and L.-M. Wang, "Bayesian network classifiers based on Gaussian kernel density," *Expert Syst. Appl.*, vol. 51, pp. 207–217, Jun. 2016.

[26] K. Ickstadt et al., "Nonparametric Bayesian networks," in *Bayesian Statistics*, vol. 9. Oxford, U.K.: Oxford Univ. Press, 2011, pp. 283–316.

[27] N. Friedman and I. Nachman, "Gaussian process networks," in *Proc. 16th Conf. Uncertainty Artif. Intell.*, 2000, pp. 211–219.

[28] J. Fox, *Applied Regression Analysis, Linear Models, and Related Methods*. Thousand Oaks, CA, USA: SAGE Publ., 1997.

[29] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*. Hoboken, NJ, USA: Wiley, 2015.

[30] J. A. Gámez, J. L. Mateo, and J. M. Puerta, "Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood," *Data Min. Knowl. Disc.*, vol. 22, no. 1, pp. 106–148, 2011.

[31] G. Schwarz, "Estimating the dimension of a model," *Ann. Stat.*, vol. 6, no. 2, pp. 461–464, 1978.

[32] D. Colombo and M. H. Maathuis, "Order-independent constraint-based causal structure learning," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3741–3782, 2014.

[33] J. Runge, "Conditional independence testing based on a nearest-neighbor estimator of conditional mutual information," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2018, pp. 938–947.
[34] E. V. Strobl, K. Zhang, and S. Visweswaran, "Approximate kernel-based conditional independence tests for fast non-parametric causal discovery," *J. Causal Inference*, vol. 7, no. 1, 2019, Art. no. 20180017.
[35] R. Salinas-Gutiérrez, A. Hernández-Aguirre, and E. R. Villa-Diharce, "Using copulas in estimation of distribution algorithms," in *Proc. Mexican Int. Conf. Artif. Intell.*, 2009, pp. 658–668.
[36] L.-F. Wang and J.-C. Zeng, "Estimation of distribution algorithm based on copula theory," in *Exploitation of Linkage Learning in Evolutionary Algorithms*. Heidelberg, Germany: Springer, 2010, pp. 139–162.
[37] A. Zhou, J. Sun, and Q. Zhang, "An estimation of distribution algorithm with cheap and expensive local search methods," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 807–822, Dec. 2015.
[38] H. Mühlenbein, J. Bendisch, and H.-M. Voigt, "From recombination of genes to the estimation of distributions II. Continuous parameters," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 1996, pp. 188–197.
[39] X. Meng, J. Li, M. Zhou, X. Dai, and J. Dou, "Population-based incremental learning algorithm for a serial colored traveling salesman problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 2, pp. 277–288, Feb. 2018.
[40] J. De Bonet, C. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1996, pp. 424–430.
[41] P. Bosman and D. Thierens, "Expanding from discrete to continuous estimation of distribution algorithms: The IDEA," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2000, pp. 767–776.
[42] S. Shakya, A. Brownlee, J. McCall, F. Fournier, and G. Owusu, "A fully multivariate DEUM algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 479–486.
[43] B. Yuan and M. Gallagher, "On the importance of diversity maintenance in estimation of distribution algorithms," in *Proc. 7th Annu. Conf. Genet. Evol. Comput.*, 2005, pp. 719–726.
[44] P. Pošík, "Preventing premature convergence in a simple EDA via global step size setting," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2008, pp. 549–558.
[45] J. Grahl, P. A. Bosman, and F. Rothlauf, "The correlation-triggered adaptive variance scaling IDEA," in *Proc. 8th Annu. Conf. Genet. Evol. Comput.*, 2006, pp. 397–404.
[46] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation* (Studies in Fuzziness and Soft Computing). Heidelberg, Germany: Springer, 2006, pp. 75–102.
[47] H. Fang, A. Zhou, and G. Zhang, "An estimation of distribution algorithm guided by mean shift," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 3268–3275.
[48] C. K. Chow and S. Y. Yuen, "An evolutionary algorithm that makes decision based on the entire previous search history," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 741–769, Dec. 2011.
[49] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.
[50] B. Gao and I. Wood, "TAM-EDA: Multivariate t distribution, archive and mutation based estimation of distribution algorithm," *ANZIAM J.*, vol. 54, pp. C720–C746, Jan. 2014.
[51] M. Pelikan and D. E. Goldberg, "Genetic algorithms, clustering, and the breaking of symmetry," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2000, pp. 385–394.
[52] J. M. Peña, J. A. Lozano, and P. Larrañaga, "Benefits of data clustering in multimodal function optimization via EDAs," in *Estimation of Distribution Algorithms*. Boston, MA, USA: Springer, 2002, pp. 101–127.
[53] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," in *Machine Intelligence and Pattern Recognition*, vol. 5. Amsterdam, The Netherlands: Elsevier, 1988, pp. 149–163.
[54] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surveys*, vol. 45, no. 3, pp. 1–33, 2013.
[55] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 71–78.
[56] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 1658–1665.
[57] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Nanyang Technol. Univ., Singapore, Rep. 201311, 2013.
[58] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," Nanyang Technol. Univ., Singapore, Rep., 2017. [Online]. Available: https://www3.ntu.edu.sg/home/epnsugan/index_files/CEC2017/CEC2017.htm
[59] B. Calvo et al., "Bayesian performance analysis for black-box optimization benchmarking," in *Proc. Genet. Evol. Comput. Conf. Compan.*, 2019, pp. 1789–1797.
[60] B. Doerr and W. Zheng, "Sharp bounds for genetic drift in estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 24, no. 6, pp. 1140–1149, Dec. 2020.
[61] W. F. Sharpe, "The sharpe ratio," *J. Portfolio Manag.*, vol. 21, no. 1, pp. 49–58, 1994.
[62] R. A. Martin, "PyPortfolioOpt: Portfolio optimization in Python," *J. Open Source Softw.*, vol. 6, no. 61, p. 3066, 2021.

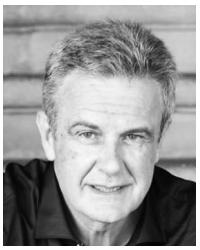
Vicente P. Soloviev received the M.Sc. degree in artificial intelligence from the Universidad Politécnica de Madrid, Madrid, Spain, in 2020, where he is currently pursuing the Ph.D. degree with the Computational Intelligence Group.

He teaches some subjects related to Artificial Intelligence for the B.Sc. in Computer Science with the Universidad Politécnica de Madrid. His research interests include the areas of probabilistic graphical models, metaheuristics for optimization, quantum machine learning, quantum heuristics, and real applications, such as industry 4.0. Vicente holds a predoctoral FPI contract awarded by the Spanish Ministry of Science and Innovation since 2021.

Concha Bielza (Senior Member, IEEE) received the M.S. degree in mathematics from the Universidad Complutense de Madrid, Madrid, Spain, in 1989, and the Ph.D. degree in computer science from the Universidad Politécnica de Madrid, Madrid, in 1996 (Extraordinary Doctorate Award).

Since 2010, she has been a Full Professor of Statistics and Operations Research with the Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid. She has published more than 150 papers in high impact factor journals and has supervised 20 Ph.D. theses. Her research interests are primarily in the areas of probabilistic graphical models, decision analysis, metaheuristics for optimization, data mining, classification models, and real applications, such as biomedicine, bioinformatics, neuroscience, and industry 4.0.

Prof. Bielza was awarded the 2014 UPM Research Prize and the 2020 Machine Learning Award from Amity University (India).



Pedro Larrañaga (Senior Member, IEEE) received the M.Sc. degree in mathematics (statistics) from the University of Valladolid, Valladolid, Spain, in 1981, and the Ph.D. degree in computer science from the University of the Basque Country (Excellence Award) in 1995.

Since 2007, he has been a Full Professor of Computer Science and Artificial Intelligence with the Universidad Politécnica de Madrid (UPM), Madrid, Spain. Before moving to UPM, his academic career developed with the University of the Basque Country, Bilbao, Spain, through several faculty ranks: an Assistant Professor from 1985 to 1998, an Associate Professor from 1998 to 2004, and a Full Professor from 2004 to 2007. He has published over 200 papers in high-impact factor journals. He has supervised over 30 Ph.D. theses. His research interests include the areas of probabilistic graphical models, metaheuristics for optimization, data mining, classification models, and real applications, such as biomedicine, bioinformatics, neuroscience, industry 4.0, and sports.

Prof. Larrañaga has received the 2013 Spanish National Prize in computer science and the prize of the Spanish Association for Artificial Intelligence in 2018 and the 2020 Machine Learning Award from Amity University (India). He is a Fellow of the European Association for Artificial Intelligence, since 2012 and the Academia Europaea, since 2018.