Information Sciences 369 (2016) 419–440

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Evolutionary algorithms and elliptical copulas applied to continuous optimization problems

Harold Dias de Mello Junior[a,*], Luis Martí[b], André V. Abs da Cruz[c], Marley M.B. Rebuzzi Vellasco[d]

[a] Electrical Engineering Department, Rio de Janeiro State University, Rio de Janeiro, Brazil
[b] Institute of Computing, Fluminense Federal University, Rio de Janeiro, Brazil
[c] Computer University Unit, State University of West Side, Rio de Janeiro, Brazil
[d] Electrical Engineering Department, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil

ARTICLE INFO

Article history:
Received 14 October 2015
Revised 28 June 2016
Accepted 2 July 2016
Available online 4 July 2016

Keywords:
Continuous numeric optimization
Copulas
Estimation of distribution algorithms
Evolutionary computation

ABSTRACT

Estimation of Distribution Algorithms (EDAs) constitutes a class of evolutionary algorithms that can extract and exploit knowledge acquired throughout the optimization process. The most critical step in the EDAs is the estimation of the joint probability distribution associated to the variables from the most promising solutions determined by the evaluation function. Recently, a new approach to EDAs has been developed, based on copula theory, to improve the estimation of the joint probability distribution function. However, most copula-based EDAs still present two major drawbacks: focus on copulas with constant parameters, and premature convergence. This paper presents a new copula-based estimation of distribution algorithm for numerical optimization problems, named EDA based on Multivariate Elliptical Copulas (EDA-MEC). This model uses multivariate copulas to estimate the probability distribution for generating a population of individuals. The EDA-MEC differs from other copula-based EDAs in several aspects: the copula parameter is dynamically estimated, using dependence measures; it uses a variation of the learned probability distribution to generate individuals that help to avoid premature convergence; and uses a heuristic to reinitialize the population as an additional technique to preserve the diversity of solutions. The paper shows, by means of a set of parametric tests, that this approach improves the overall performance of the optimization process when compared with other copula-based EDAs and with other efficient heuristics such as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES).

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

Optimization concepts and tools have been widely used in quantitative methods to address extensive classes of decision support problems, from various fields of knowledge, especially in engineering. In the absence of computationally treatable deterministic solution proposals, stochastic algorithms become a viable alternative for their high chances of identifying a good solution and even of reaching the global optimum in a reasonably short time. Among the many stochastic global

* Corresponding author.
E-mail address: harold.dias@gmail.com (H.D. de Mello Junior).

http://dx.doi.org/10.1016/j.ins.2016.07.006
0020-0255/© 2016 Elsevier Inc. All rights reserved.

optimization algorithms [57,62], some are nature-inspired metaheuristics. The most well-known ones are swarm intelligence and evolutionary algorithms.

Swarm intelligence algorithms take inspiration from the collective behavior of social insects and other animal societies [5]. These algorithms assume a population of simple individuals/agents capable of jointly producing a self-organized behavior for their survival. The most successful swarm intelligence inspired optimization algorithms are ant colony (ACO) [13] for discrete problems, and particle swarm optimization (PSO) [29] for continuous problems, including real-world tasks.

Evolutionary algorithms, such as genetic algorithms [38], are also population-based metaheuristics of great research interest because of their promising results in different applications, such as telecommunications [27]. These metaheuristics use principles of Darwin's theory of natural selection: at each generation or iteration of the algorithm, a competitive selection occurs to choose the best solutions; these are modified by operators to generate new solutions, repeating this cycle until a given stop criterion, defined by the user, is reached.

In the same group of population-based metaheuristics are the Estimation of Distribution Algorithms (EDAs). EDAs [25,42] exploit promising regions of the search space by sampling the probabilistic model, which is estimated previously and periodically from a set of candidate solutions selected by the algorithm according to their fitness values. This learning mechanism and the statistical sampling allow EDAs to detect explicitly the dependence relationship between decision variables, transferring it to the new solutions of the next generation, and to have better performance than other evolutionary algorithms since the optimization uses, in general, a smaller number of objective function evaluations [33].

The ability to solve large and complex problems with superior performance than other global optimization techniques [25] has been a great motivating factor for the development of new approaches to this type of algorithm, all of them seeking to add estimation methods with the best trade-off between accuracy on the one hand and computational cost of learning and sampling of the probabilistic model on the other. In the recent years, different techniques have been proposed to improve its estimation [25,31], including but not limited to models based on copula theory. Copulas, according to Sklar's theorem [53], are used to build a joint probability distribution relating the dependence structure (established by a copula) to its marginal univariate distributions, instead of assuming unrealistic Gaussian distributions as a proxy for the joint probability distribution. This technique reduces the computational effort for model learning and gives greater flexibility in modeling multivariate data, generating important contributions in various applications, such as financial risk management [11]. In the context of evolutionary computing, the use of copulas in EDAs is more recent and therefore the subject of much research interest. Copula-based EDAs [20] start by calculating the marginal distribution of each variable separately. They then pick a particular copula to construct the joint distribution. Given the marginal distributions and a copula, it is then possible to generate new candidate solutions. Despite the good results achieved in tests with some numerical optimization benchmarks, the current copula-based EDAs present two major limitations. First, the copula parameters are assumed to be constant, which reduces the algorithm's performance. The second limitation concerns premature convergence to a local optimum. Although the capacity of the class of probability distribution used in the EDAs has grown significantly, real-valued copula-based EDAs tend to converge prematurely on problems such as the Rosenbrock problem [9]. Moreover, aspects of modeling copulas in EDAs need still to be investigated, including the limitations of building multivariate copulas, efficiency in the relationship between the type of copula and the marginal distributions, and mechanisms to maintain the population diversity to make it a competitive tool for global optimization.

This paper presents a new approach to copula-based EDAs that is more efficient than other similar algorithms and very promising when compared with one of the best evolutionary algorithms for numerical optimization problems, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [21,22]. The proposed approach, named EDA based on Multivariate Elliptical Copulas (EDA-MEC), is based on elliptical copulas, such as the multivariate Gaussian and Student's t copulas. They are widely used in empirical applications due to their tractable properties similar to those of multivariate normal distributions. EDA-MEC considers the construction of probabilistic models with a better trade-off between accuracy and computational effort and differs from the others in several respects: the copula parameters are updated dynamically in every algorithm generation, generating more reliable search distributions; the approach makes use of variations of the estimated probabilistic model to insert diversity into the population, avoiding premature convergence; and it uses an adaptive heuristic to reinitialize the population throughout an elitist evolution as an additional mechanism for diversity.

The structure of the paper is the following: Section 2 presents the mechanisms and characteristics of the main evolutionary algorithms, including EDAs. Section 3 provides a brief introduction to copulas and copula-based EDAs. Section 4 provides details of the proposed model, the Estimation of Distribution Algorithm based on Multivariate Elliptical Copulas (EDA-MEC). Section 5 presents the experimental setting to solve a test suite for numerical optimization, compares the EDA-MEC performance to other models, and discusses the results. Finally, Section 6 articulates some general conclusions.

2. Fundamentals of evolutionary algorithms

Evolutionary computation is a subfield of computational intelligence, covering various techniques inspired by biological evolution mechanisms; these are considered stochastic methods of global optimization. Most of these methods use population-based metaheuristics [39,57]. Algorithms such as evolutionary algorithms, collective intelligence algorithms (swarm computing) and memetic algorithms belong to this class of metaheuristics.

The most significant advantage of evolutionary computing is the robustness and flexibility of its algorithms based on generic and adaptable procedures, which allow solving complex problems for which traditional methods are ineffective.

Based on genetic inheritance and the evolution theory, evolutionary algorithms usually start from a population of randomly generated solutions. Then, after a selection mechanism based on fitness, “parent” solutions are chosen to produce “offspring” solutions via a reproductive plan, under which genetic operators such as crossover and mutation are applied to the "parent" solutions. Finally, the new solutions are incorporated into the original population, replacing some or all of the old ones, and a new cycle of evolution begins until one or more stop criteria previously defined are reached.

During the past 40 years, a variety of evolutionary algorithms has been proposed in the literature, of which the major ones are: (i) Genetic Algorithms (GAs) [38]; (ii) Evolutionary Programming (EP) [16]; (iii) Evolution Strategies (ESs) [49]; (iv) Classifier Systems (CSs) [7]; and (v) Genetic Programming (GP) [30]. EDAs [25,42], which constitute a variation on genetic algorithms, are also considered evolutionary algorithms, following the same operational flow as the others. Evolutionary algorithms have been widely used in science and engineering for solving complex problems. Xiangman and Lixin [61] propose an intelligent particle filter and its application to fault detection of nonlinear system and presents a comprehensive description of evolutionary algorithms.

The difference between the evolutionary algorithms is in the way they represent the data, the kinds of genetic operators used and the individual selection mode. Evolution Strategies (ESs) [49], which are particularly reliable and highly competitive evolutionary algorithm for black box optimization, represent data as real-valued vectors and exploit the ideas of adaptation and evolution. A vector of strategy parameters is evolved along with the problem-solution vector in a process called self-adaptation, which is critical to the success of the ES in optimization tasks. There is usually a strategy parameter for each problem-solution variable; the strategy parameters are used to scale the standard deviation of the mutation distribution that will be applied to the variable. Then, that better strategy parameters will tend to be selected, and the variance of the mutation distribution will be optimal. This allows the ES to be self-optimizing in its performance, to the extent that the genes can mutate at different rates, making the ES an ideal optimization method for dynamic functions. With respect to genetic operators, mutation and selection are present in all of the ESs, and crossover is rarely used.

One of the most prominent ESs, which is one of the current state of the art methods in continuous optimization problems, is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), proposed in [21]. The outstanding performance of this approach lies in the direct and nonrandom way in which it updates the strategy parameters. Information obtained from the evolution path −steps carried out by the algorithm in d iterations− guides the intensity of changes in the covariance matrix: increasing when variations in the direction of space are successful and gradually decreasing in other directions. Cumulative information about successive steps makes it possible to reliably adjust the covariance matrix even when the algorithm uses small populations. Another important property of CMA-ES is its invariance against linear transformations of the search space, including rotation, reflection and translation. Hansen et al. [22] argues that invariance properties are desirable in optimization algorithms, because they increase the predictive power of performance results by inducing problem equivalence classes. Furthermore, all experimental comparisons confirm the superiority of this approach over the purely mutative evolutionary strategies.

CMA-ES has been extended from its original formulation in order to further enhance its performance and allow it to deal with complex constraints imposed to the optimization problem. One of such approaches consists on the incorporation of CMA-ES in a memetic structure [41]. In these cases, CMA-ES is complemented by local search methods. For instance, in [58] it is studied the combination of CMA-ES with a local optimizer that takes care of refining the best solution by CMA-ES after every iteration. Another form of carrying out this combination is to resort to the Separability Prototype for Automatic Memes (SPAM) [10], which proposes a software platform for the automatic design of search methods in continuous optimization. SPAM maintains two separate algorithms running in parallel, each meant for handling separable and non-separable aspects of the problem being solved, respectively. Subsequently, [15] proposed a modified SPAM with hyper-heuristic coordination of the operators, namely SPAM-AOS, and showed that the Adaptive Operator Selection (AOS) appears to improve upon the original memetic computing. Another interesting approach is put forward by [35] in which CMA-ES takes part of a memetic structure that enhances its capacity for dealing with constraints in the optimization process.

By combining ideas from machine learning with evolutionary algorithms, estimation of distribution algorithms address a broad class of problems, outperforming other optimization techniques. Hence, the next section provides a brief introduction to EDAs and the main classes of these algorithms, including copula-based EDAs, which are the essence of the proposed model.

2.1. Estimation of distribution algorithms (EDAs)

EDAs share some working principles with genetic algorithms (GAs). However, in EDAs, new (offspring) individuals are generated in a different way. That is, instead of using genetic recombination and mutation operators, EDAs estimate the probability distribution of the most promising solutions and then sample it to generate new solutions. This mechanism gives the EDAs a feature non-existent in GAs: the ability to extract and use knowledge throughout the execution.

Another important difference is the way solutions are built. While GAs try to implicitly combine the building blocks, EDAs explicitly seek to find the correlation or relationship of dependency between variables. This is the main advantage of EDAs over other evolutionary algorithms: the ability to treat robustly (with little variation in performance between runs) and scalable problems containing important interactions between variables [34,45].

There are other approaches that share population modeling principle of EDAs but have emerged from other contexts of the evolutionary computation field, and therefore have been studied and presented in a separate form. That is the case of the Compact Differential Evolution (cDE) algorithm [40]. cDE maintains a statistic description and, in addition, also applies the mutation and crossover typical of differential evolution (DE), thus reproducing its search procedure. Another similar approach is that of the compact Particle Swarm Optimization (cPSO) algorithm [44]. cPSO employs the search procedure of Particle Swarm Optimization (PSO) algorithms, but unlike classical approaches, does not use a swarm of particles and their corresponding positions and velocities. Instead of that, cPSO constructs a probabilistic representation of the swarm's behavior.

The fundamental step of EDAs is to estimate the joint probability distribution of the input variables of solutions. This task becomes more difficult when the problem at hand has complex dependencies between variables, requiring a trade-off between accuracy and computational estimation of cost. Hence, the literature usually classifies EDAs according to the complexity of the probabilistic models used to capture the interdependencies between the variables: no dependencies, pairwise dependencies, multivariate dependencies and models with mixture distributions [31].

The first category (i.e., no dependencies) is by far the simplest. All the algorithms in this category compute the joint probability distribution as a product of univariate distributions, $p(X) = \prod_{i=1}^{n} p(X_i)$, where X is the set of promising solutions in an n-dimensional search space. Population-Based Incremental Learning (PBIL) [3], the Univariate Marginal Distribution Algorithm (UMDA) [42], and the Compact Genetic Algorithm (cGA) [23] are univariate discrete EDAs; UMDAc [32] is an adaptation of the UMDA to continuous domains. Although these algorithms are simple, fast and highly scalable, they ignore feature dependencies and exhibit bad performance on deceptive problems. These limitations brought up the need for a more advanced and capable probabilistic modeling.

The assumption of the pairwise-dependencies models is that a variable depends only on one other variable and the joint probability distribution is expressed by $p(X) = \prod_{i=1}^{n} p(X_i|X_{j_{(i)}})$, with $i \neq j$ and where $X_{j_{(i)}}$ is the variable that X_i is dependent on. Examples in this category include Mutual Information Maximizing Input Clustering (MIMIC) [6], Combining Optimizers with Mutual Information Trees (COMIT) [4], and the Bivariate Marginal Distribution Algorithm (BMDA) [47].

Although pairwise-dependencies models can be effective in certain scenarios, in multivariate problems they are insufficient to adequately represent the solution space. In these cases, all dependencies between variables are modeled by a multivariate distribution, as follows: $p(X) = \prod_{i=1}^{n} p(X_i|X_1, X_2, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k)$. There are examples of EDAs with multivariate dependencies: the Bayesian Optimization Algorithm (BOA) [46], the Markov Network Estimation Distribution Algorithm (MN-EDA) [52], the Hierarchical Bayesian Optimization Algorithm (hBOA) [45], and the Extended Compact Genetic Algorithm (ECGA) [24].

Besides the basic probabilistic models, EDAs have also used other modeling techniques. Models with mixture distributions have increased flexibility in estimating the joint probability distribution on EDAs, especially for cases of multimodal optimization problems. A mixture probability distribution is a weighted sum of probability distributions, each of which is a function of all random variables. These models are described by $p(X) = \sum_{i=1}^{k} \pi_i p_i(X)$, where π_i represents the weight of the ith component of the mixture; $p_i(x)$ is the probability distribution of the ith group, created by the data clustering method; and k is the number of mixture components. However, models with mixture distributions have some challenges, such as definition of the shape of each component of the mixture, determination of the number of components, and diversity maintenance. When they are implemented, there is a concern about the computational cost of the learning algorithms for the parameters of the mixture model. This approach is, therefore, computationally heavy and the more complex model is, the higher the necessary computational time to find the best solution. There are EDAs with mixture distributions: the Estimation of Mixtures of Distributions Algorithm (EMDA) [48], the mixed Iterated Density Estimation Evolutionary Algorithm (mIDEA) [8], and the Online Gaussian Mixture Model Algorithm (EDAOGMM) [18].

Therefore, the results obtained from the EDA family of algorithms are not yet competitive compared to more traditional metaheuristics [57]. This has motivated the development of new algorithms. More recently, some EDAs have employed copula theory [53] to solve real-valued optimization problems and relax the normality assumption for the variables. Copula-based EDAs use only a copula function and the marginal univariate probabilities to estimate the joint probability distribution. This reduces the computational complexity of model learning. A brief introduction to copula theory and copula-based EDAs is presented in the next section.

3. Copula theory and copula-based EDAs

3.1. Copula theory

The concept of copulas was introduced by Sklar (1959) [53], but it was applied to finance only many years later [11]. The use of copulas makes it possible to separate the dependence model from the marginal distributions. Essentially, copulas provide a straightforward way to extend financial modeling from the usual joint normality assumption to more general joint distributions, preserving the normality assumption on the marginal. Thus, copulas constitute a very useful tool for allowing the construction of multivariate models that adequately characterize linear and nonlinear dependencies between the variables of a problem. Mathematically, copulas are based on the following definition:

Definition 3.1. (Copula)

Let $u = (u_1, \ldots, u_n)$ with $u_i \in [0,1]$ be a vector of n variables. A function $C(u): [0,1]^n \to [0,1]$ is an n-dimensional copula if and only if it satisfies the following basic conditions:

- $C(u) = u_k$ if all coordinates of u are 1, except u_k;
- $C(u) = 0$ if at least one of the coordinates of u is zero; and
- C is increasing in each coordinate of u.

The main result in the theory of copulas is the celebrated Sklar's theorem, which is recounted below.

Theorem 3.1. *(Sklar)*

According to Sklar's theorem, a Joint Cumulative Distribution Function (JCDF) F of random variables $X_1, X_2, \ldots, X_n$, with continuous marginal distributions $F_1, F_2, \ldots, F_n$, respectively, can be characterized by a single n-dimensional dependency function or copula C, such that for all vectors $\boldsymbol{x} \in \overline{\mathbb{R}^n}$:

$$\begin{aligned} F(x_1, x_2, \ldots, x_n) &= C(F_1(x_1), F_2(x_2), \ldots, F_n(x_n)) \\ &= \mathrm{P}\{X_1 \le x_1, X_2 \le x_2, \ldots, X_n \le x_n\} \end{aligned} \tag{1}$$

Furthermore, Sklar's theorem asserts that if all the marginals $F_1, F_2, \ldots, F_n$ are continuous, then the copula function C is uniquely defined. In this case, the univariate margins can be separated from the multivariate dependence structure, which is represented by a copula. On the other hand, when the marginal distributions are discrete, it is not possible to assume that the copula function of Eq. (1) is unique.

As an alternative representation, Eq. (1) can be written in the inverted form. For any vector $\boldsymbol{u} \in [0,1]^n$:

$$C_F(u_1, u_2, \ldots, u_n) = F\left(F_1^{-1}(u_1), F_2^{-1}(u_2), \ldots, F_n^{-1}(u_n)\right) \tag{2}$$

where C_F denotes the copula associated with F and $F^{-1}(\boldsymbol{u}) = \inf\{x \in \mathbb{R} \mid F_i(x_i) \ge \boldsymbol{u}\}$, with $i = 1, \ldots, n$, constitutes the generalized inverse function of F. Then $x_1 = F_1^{-1}(u_1) \sim F_1, \ldots, x_n = F_n^{-1}(u_n) \sim F_n$. It follows that an n-copula is a multivariate distribution with all n univariate margins being $U(0,1)$.

Theorem 3.2. *(Partial derivative of copulas)*

Let C be a copula [43]. For any $u_n \in [0,1]^n$, the partial derivative $\frac{\partial}{\partial u_n} C(u_1, u_2, \ldots, u_n)$ exists for almost all u_n, and for such $u_1, u_2, \ldots, u_n$,

$$0 \le \frac{\partial}{\partial u_n} C(u_1, u_2, \ldots, u_n) \le 1 \tag{3}$$

Definition 3.2. (Joint probability density function)

From Eq. (1) and the previous theorem, we can define the joint probability density function (JPDF), f, as the mixed nth-order derivative of copula C, i.e.:

$$\begin{aligned} f(x_1, \ldots, x_n) &= \frac{\partial^n C(F_1(x_1), \ldots, F_n(x_n))}{\partial F_1(x_1) \cdots \partial F_n(x_n)} \prod_{i=1}^{n} f_i(x_i) \\ &= c(F_1(x_1), \ldots, F_n(x_n)) \prod_{i=1}^{n} f_i(x_i) \end{aligned} \tag{4}$$

where F_i, f_i are the distribution functions and the marginal and

$$c(F_1(x_1), \ldots, F_n(x_n)) = \frac{\partial^n C(u_1, \ldots, u_n)}{\partial u_1 \cdots \partial u_n} = \frac{f(x_1, \ldots, x_n)}{f_1(x_1) \cdots f_n(x_n)} \tag{5}$$

Eq. (4) presents the clear decomposition of the joint density $f(x_1, \ldots, x_n)$ into two parts: one that only describes the dependency structure and one that describes the marginal behavior of each variable. In practice, this decomposition simplifies the specification of the multivariate distribution and its estimation.

Since copulas can capture dependence structures regardless of the form of the margins [11,43], copulas provide a natural way to study and measure dependence between random variables. It follows that measures of association and positive dependence concepts are properties of copulas [43]. Linear correlation, or Pearson's correlation, is the most frequently used in practice and a natural measure of dependence for elliptical distributions, especially the multivariate Gaussian and Student's t distributions. Outside the world of elliptical distributions, however, using the linear correlation coefficient as a measure of dependence may lead to misleading conclusions given that it is not a robust measure of nonlinear dependence cases [14]. Thus, other measures should be considered, such as copula-based ones. One such measure is tail dependence, which is related to the amount of dependence in the upper-right-quadrant tail or lower-left-quadrant tail of a bivariate distribution. [43] demonstrates that the tail dependence coefficients λ_U and λ_L are non-parametric and depend only on the copula C of random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$. Consequently, it is invariant under strictly increasing transformations of the random variables.

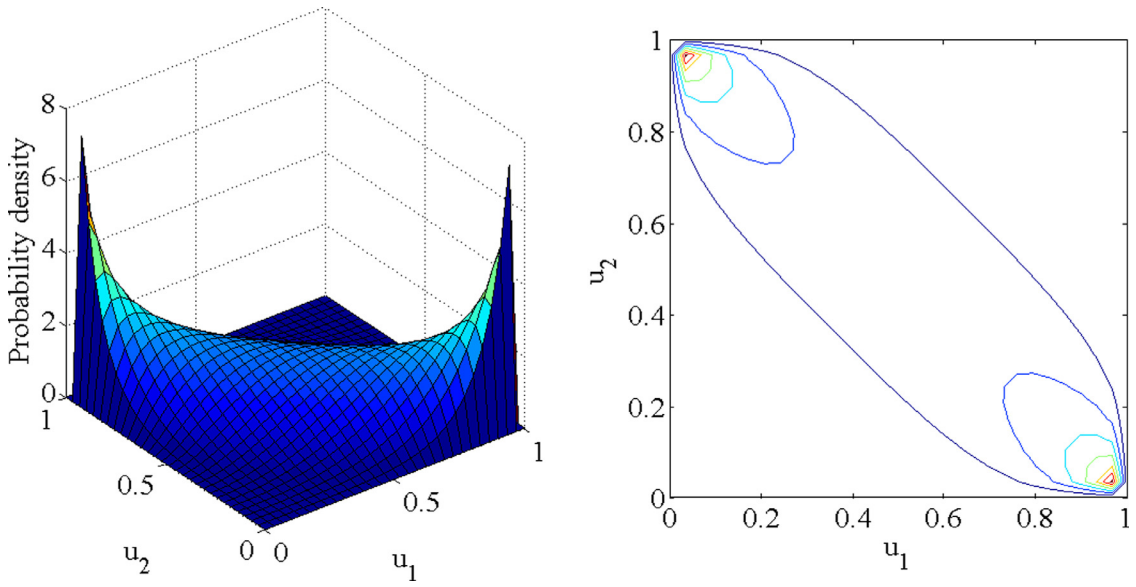

Fig. 1. Density and contour of density for Gaussian copula with $\rho = -0.8$, where there is no tail dependence.

Although there is a large variety of copulas [28,43], only a few are treatable, and often the choice in dependence modeling falls on an elliptical copula. Its analytical form is obtained via Sklar's theorem, specifically when the distribution function F in Eq. (2) is elliptical. Thus, elliptical copulas also have symmetric tail dependence. Elliptical distributions are generalizations of the multivariate normal distribution and, therefore, share many of its tractable properties, allowing one to simulate elliptical distributions as a linear transformation of standard elliptical distributions.

It is possible to obtain the general expression for the multivariate Gaussian copula by using Eq. (2):

$$C_R^{\mathrm{Ga}}(\boldsymbol{u}) = \Phi_R^n\left(\Phi^{-1}(u_1), \ldots, \ \Phi^{-1}(u_n)\right) \tag{6}$$

where Φ_R^n is the copula of $\boldsymbol{X} \sim \mathcal{N}_d(0, R)$ – the standard multivariate normal distribution with linear correlation matrix $R_{ij} = \Sigma_{ij}/\sqrt{\Sigma_{ii}\Sigma_{jj}}$ – and Φ^{-1} is the inverse (quantile function) of the univariate normal distribution. In the bivariate case, Eq. (6) reduces to:

$$C_R^{\mathrm{Ga}}(u_1, u_2) = \int_{-\infty}^{\phi^{-1}(u_1)} \int_{-\infty}^{\phi^{-1}(u_2)} \frac{1}{2\pi\left(1 - R_{12}^2\right)} \exp\left\{-\frac{x^2 + y^2 - 2R_{12}xy}{2\left(1 - R_{12}^2\right)}\right\} dxdy \tag{7}$$

where $R_{12} = \rho$ is the linear correlation coefficient. The radial symmetry of the Gaussian copula expressed by the equality of tail dependence coefficients can be seen in Fig. 1.

With $\rho = 0$, we obtain a very important special case of the Gaussian copula, which takes the form $C(u_1, u_2) = u_1 u_2$ and is called the product copula; this is related to the fact that two variables are independent if and only if their copula is the product copula.

On the other hand, the Student's t copula introduces an additional parameter (degree of freedom), allowing for joint fat tails and an increased probability of joint extreme events compared with the Gaussian copula. The general expression for the Student's t copula can also be obtained by using Sklar's theorem:

$$C_{\nu,\ R}^{\mathrm{t}}(\boldsymbol{u}) = t_{\nu,\ R}^n\left(t_\nu^{-1}(u_1), \ldots, \ t_\nu^{-1}(u_n)\right) \tag{8}$$

where:

ν is the degrees of freedom of the distribution;

$R_{ij} = \Sigma_{ij}/\sqrt{\Sigma_{ii}\Sigma_{jj}}$, with $i, j \in \{1, \ldots, n\}$;

$t_{\nu,\ R}^n$ is the distribution of $\sqrt{\nu}\mathbf{Y}/\sqrt{S}$, where $S \sim \chi_\nu^2$ and $\mathbf{Y} \sim \mathcal{N}_n(0, R)$ are independent; and

t_ν denotes the marginal of $t_{\nu,R}^n$, i.e., the distribution of $\sqrt{\nu}Y_1/\sqrt{S}$.

The bivariate Student's t copula is given by:

$$C_{\nu,R}^{\mathrm{t}}(u_1, u_2) = \int_{-\infty}^{t_\nu^{-1}(u_1)} \int_{-\infty}^{t_\nu^{-1}(u_2)} \frac{1}{2\pi\left(1 - R_{12}^2\right)^{1/2}} \exp\left\{1 + \frac{x^2 - 2R_{12}xy + y^2}{\nu\left(1 - R_{12}^2\right)}\right\}^{-\frac{(\nu+2)}{2}} dxdy \tag{9}$$

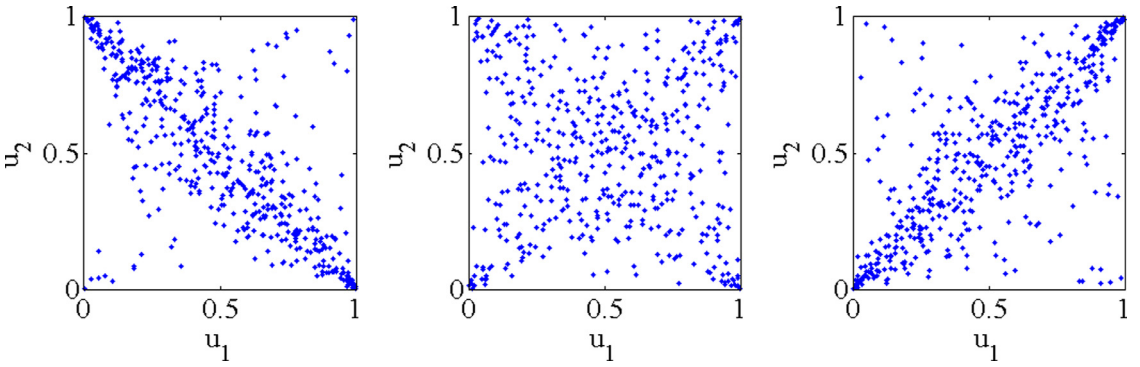

Fig. 2. Scatterplot of 500 samples from the bivariate Student's t copula ($\nu = 1$) with $\rho = -0.8$ (left, negative dependence), $\rho = 0$ (middle, independence) and $\rho = 0.8$ (right, positive dependence). The scatterplot has two axes of symmetry: (1) symmetry around the first diagonal (corresponding to exchangeability: $C(u_1, u_2) = C(u_2, u_1)$) and (2) symmetry around the second diagonal (corresponding to the fact that the Student's t copula is radially symmetric).

where $R_{12} = \rho$ is the linear correlation coefficient. When the number of degrees of freedom increases, the copula converges to the Gaussian one. For a limited number of degrees of freedom, however, the behavior of the two copulas is quite different. Scatterplots of the Student's t copula are provided in Fig. 2.

Computational aspects are one of the main topics from a modeling point of view. This concerns the copula parameter estimation problem and the simulation issue. If the two problems cannot be easily solved for high dimensions for a given copula, the copula is not tractable to estimate the multivariate distribution. The estimation of the ρ parameter of the Gaussian and Student's t copula is very easy. However, the Student's t copula generally requires an optimization algorithm to obtain the degrees of freedom. Moreover, simulations based on both copulas are straightforward when the samples are easily obtained even if an elliptical copula is not in closed form [11].

3.2. Copula-based EDAs

Given the ability of copulas to capture the entire dependence structure from a set of multidimensional random variables – modeling important problems in very different applied fields – it is quite reasonable to associate them with EDAs, through Copula-based Estimation of Distribution Algorithms (CEDAs), which consequently can build search distributions that are more realistic than other EDAs. Basically, copulas act in the learning step of EDAs, i.e., after selecting the best individuals: estimating first the marginal distributions and then the dependence structure.

According to Sklar's theorem, any multivariate distribution function can be represented as a copula of its univariate marginals. Since a copula completely determines the dependence structure of variables and the multivariate distribution F is obtained by (1), the issue of estimating the probabilistic model of a copula-based EDA is split into two parts: (i) selecting or building the copula C; (ii) estimating each univariate marginal F_i for $i = 1, \ldots, n$. After estimating the probabilistic model, the next step is sampling it to generate the new solutions. In order to do this, the algorithm generates a random vector $\{u_1, u_2, \ldots, u_n\} \in [0, 1]^n$ that follows C (basic property of the copula theory, presented in Definition 3.1). Next, the x_i values are calculated by the inverse function of their corresponding marginal, i.e., $x_i = F^{-1}(u_i)$, thereby generating a new individual $\{x_1, x_2, \ldots, x_n\}$, which is a sample that follows the multivariate distribution F.

The use of copulas in evolutionary algorithms is very recent. The first attempt to associate copulas with EDAs was a technical report [54]. Since then, a considerable number of CEDAs have been proposed in the literature. For recent survey the reader is referred to [20].

Identified as an emerging approach for the solution of real-valued optimization problems [25], CEDAs usually assume a particular distribution (e.g., normal or beta) for each margin, and its parameters are estimated by maximum likelihood. In other cases, kernel density estimation or empirical marginal distributions have been used.

In recent years, few new CEDAs have been produced. In addition to the aforementioned, the following are identified in the literature: a hybrid algorithm of Differential Evolution [55] with an empirical CEDA for dynamic optimization problems [60]; a Gaussian CEDA with model migration applied to numerical optimization benchmarks [26]; the use of graphical models and copula functions in EDAs for solving multivariate optimization problems [51]; and a multivariate Archimedean CEDA whose parameters are estimated from the current population by means of Kendall's tau and applied to multiobjective problems [17]. Archimedean copulas are a particular class of copula that do not derive directly from Sklar's theorem. Although Archimedean copulas have interesting properties, this work has concentrated on elliptical copulas, since they offer a good compromise between convenience and flexibility: it is easier to generalize them to a higher number of dimensions than Archimedean copulas.

In 2014, [20] implemented five models of multivariate CEDAs in the copulaedas package for the R language. These models will be used to evaluate the proposed algorithm's performance:

- UMDA (Univariate Marginal Distribution Algorithm) for continuous variables. Although originally it was not defined in terms of copulas, this algorithm is a CEDA that models the dependence structure between the variables using a multivariate product copula. Therefore, it assumes random variables are independent.
- GCEDA (Gaussian Copula Estimation of Distribution Algorithm). This algorithm is based on the multivariate normal copula, which allows the construction of multivariate distributions with a normal dependence structure and non-normal margins.
- CVEDA (C-Vine Estimation of Distribution Algorithm) and DVEDA (D-Vine Estimation of Distribution Algorithm). These models use graphical models, called regular vines, Canonical and Drawable, which allow the building of multivariate distributions using only bivariate copulas (pair-copulas) with different types or strengths of dependence [1]. The implementation of CVEDA and DVEDA included in the copulaedas package uses the elliptical (Gaussian and Student's t) and the Archimedean copulas (Clayton, Frank and Gumbel).
- Copula MIMIC. This is an extension of MIMIC (Mutual Information Maximization for Input Clustering) for continuous domains; it learns a chain dependence structure, but it uses bivariate copulas instead of bivariate normal distributions to model dependences.

4. The proposed model – EDA-MEC

The main reason for proposing a copula-based EDA is the flexibility that copulas provide for building effective joint distributions of the search space, instead of assuming that the variables follow a Gaussian distribution, which is common in most EDAs. This assumption often leads to inconsistent results and, therefore, the poor performance of the algorithm, especially in problems of numerical optimization in multivariate spaces. Thus, the use of copulas in EDAs can simplify their operation, once the construction of the probability models is performed in less time and with greater accuracy than other estimation techniques.

As most copula-based EDAs, EDA-MEC is composed of the following 4 modules:

(1) Selection or construction of the copula;
(2) Estimation of the copula parameter(s);
(3) Estimation of the marginal distributions; and
(4) Sampling the copula and generating the new individuals.
Additionally, the proposed model has two new modules related to the maintenance of diversity and correction of the correlation between the variables:
(5) Transforming the probabilistic distribution to generate rebel individuals; and
(6) The population restarting.

The first four modules are explained below, while the new extra modules are presented in the next subsection. All six modules are illustrated in Fig. 3.

As aforementioned, the most important step and the bottleneck of EDAs is the estimating the joint probability distribution that is associated with the variables from the most promising solutions found by the evaluation function. By using copulas, the EDA-MEC seeks to simplify this difficulty. First, copulas are a powerful statistical tool that isolates the dependence structure between variables. Thus, by linking together any number of marginal distributions, a copula is capable of generating a valid multivariate distribution with much less computational effort and greater accuracy than any classical estimation technique. Second, by using elliptical copulas, tractable calculus like the multivariate normal distribution are permitted, that is, building multivariate distributions with elliptical copulas, such as Gaussian and Student's t copulas, is very easy to EDA-MEC. Moreover, the dependence between variables and the accuracy of EDA-MEC is updated in each generation with the correlation of the current population. Finally, the problem of premature convergence of EDAs is improved with EDA-MEC, by the addition of modules 5 and 6, transforming the probabilistic distribution to generate rebel individuals and population restarting, respectively.

The first module of the proposed model, the Estimation of Distribution Algorithm based on Multivariate Elliptical Copulas (EDA-MEC), consists of defining the type of copula to be used in the optimization. Currently, the user can choose between two classical elliptical copulas:

- Gaussian
- Student's t

Although these copulas cannot be expressed in a simple closed form, they are tractable for high dimensions, and multivariate samples can be obtained easily. Furthermore, they are the most widely known and applied copulas in other areas of data modeling, such as finance [11] and hydrology [50].

With respect to the second module, the main parameter for both the Gaussian copula and the Student's t copula is the linear correlation matrix R between random variables represented by the population. In addition, the Student's t copula has one more parameter to be estimated: the ν degrees of freedom. This is a shape parameter that controls the kurtosis of the

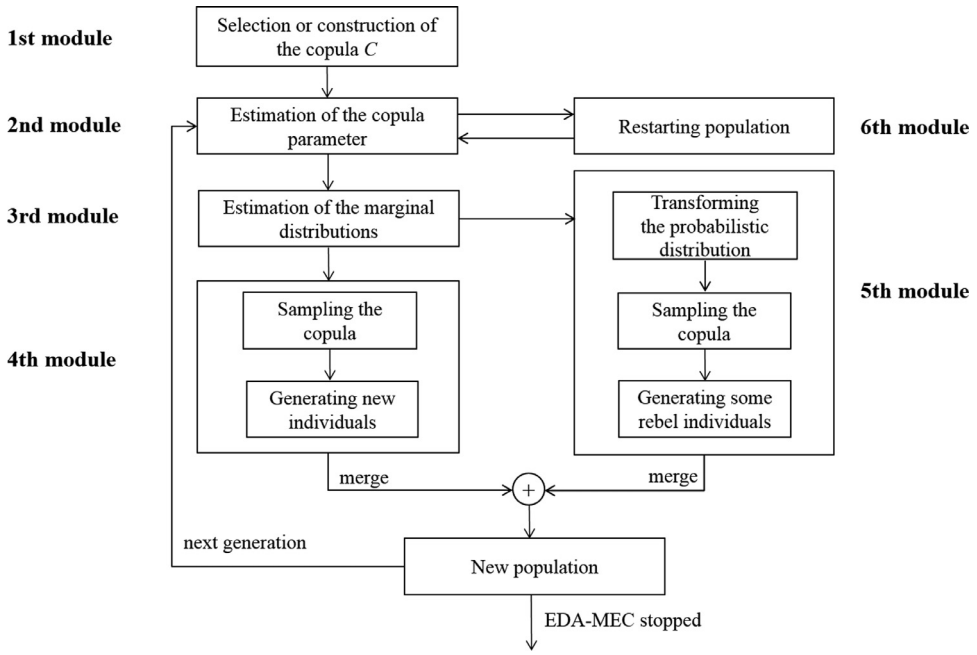

Fig. 3. EDA-MEC simplified flowchart.

distribution: the higher the degrees of freedom, the closer is the behavior of a multivariate Student's t distribution to a multivariate Gaussian distribution. Typically, the degrees of freedom are computed using Maximum Likelihood Estimators (MLEs), holding the copula correlation parameters fixed. In the case of EDA-MEC, to reduce the computational cost of the MLE, when the user selects the Student's t copula, the number of degrees of freedom is fixed and equals the quantity of new individuals, i.e., equal to the sample size, and the linear correlation matrix R is updated at each iteration of the algorithm, using the current population as the data inputs. Given that the case studies (next section) used populations of up to 30 individuals, the tests assumed a Student's t copula with a low number of degrees of freedom. Consequently, during the trials the Student's t copula never converged to the Gaussian copula. However, with this model parameterization, the EDA-MECs are less likely to generate random samples from the extreme value distribution, which can affect its performance. In order to verify such implications (in the next section), performance comparisons were made of the EDA-MEC using the Gaussian copula and the Student's t copula having identical parameterization and the degrees of freedom estimated by MLE.

Regarding the third module, according to Sklar's theorem, expressed in Eqs. (1) and (2), if the marginal distributions $F_1, F_2, \ldots, F_n$ were known, samples of $u_i = F_i(x_i)$ would be drawn from the copula. However, the margins are unknown, needing to be estimated parametrically or non-parametrically. EDA-MEC testing was performed with two types of marginal distributions: a parametric (normal), and a non-parametric (empirical). This considers the pseudo-observations, since this approach does not make any assumptions about the variables' dependency structure.

In this way, in practice, the inverse functions of the marginal distribution function are estimated and not the marginal distributions themselves. This is strictly the application of Sklar's theorem in reverse order (see Eq. (2)), which, combined with the straightforward definition (see Eq. (1)), allows us to understand that:

$$x_i = F_i^{-1}(u_1) \tag{10}$$

where u_i, x_i and F_i^{-1} are as previously defined. This process can be applied to any kind of marginal distribution.

Recalling that the Cumulative Distribution Function (CDF) of a random variable X is a non-decreasing function of x and right-continuous, we can see that the generalized inverse of a distribution function F, $F^{-1}(u)$, is therefore the quantile function of F; the inverse of the empirical marginal is determined thereby. For the normal distribution, its inverse is obtained by the quantile function of a standard normal distribution, that is, with a mean of 0 and standard deviation of 1.

Finally, in the fourth module, samples are generated from the elliptical copulas. EDA-MEC uses the algorithms presented in [11,37] for the simulation of random variables with copulas. Samples of u_i were drawn from the Gaussian and Student's t copulas and using Eq. (10) the inverse CDF of these samples was taken to retrieve the corresponding values of x_i, the new individuals of EDA-MEC.

Next, the fifth and the sixth modules, proposed in this work, are described below.

4.1. Population diversity techniques

Although EDAs have been extremely efficient in solving problems that are difficult even for other evolutionary techniques, this type of algorithm has poor performance in some cases, such as in multimodal multi-objectives and non-stationary problems, notably because of the very rapid convergence along with the lack of operators that can either maintain or reintroduce diversity into the population. This phenomenon is known as premature convergence, and it entails a risk of stagnation of the process in local optima. Hence, some proposals can be found in the literature that try to preserve the population diversity. The main techniques use various subpopulations or niches [36], each one containing solutions belonging to a different region of the search space, during the evolutionary process.

Wallin and Ryan [59] introduced a technique based on "sampling-mutation" and showed that it increases the performance of a non-copula-based EDA on a non-stationary problem and a hierarchical problem. In that study, a mutation operator is defined that changes the probabilities of a certain variable by its complement when sampling the model. Instead of mutating the probability of the variables randomly and generating individuals similar to those already contained in the population, [12] proposed another technique, in which a new probability distribution is obtained as a variation of the learned one so as to generate individuals with a lower probability of appearing in the evolutionary process. These individuals were named "rebel individuals".

In the fifth module, the EDA-MEC, immediately before the increment of the iteration, replaces some of the individuals with the lowest evaluation by a variation of the original definition of "rebel individuals". These are also sampled from a modification of the estimated probabilistic model at each iteration, but that is not necessarily its inverse, since the EDA-MEC builds a CDF, and [12] builds a Probability Density Function (PDF). The EDA-MEC builds a new model from the complement of the samples u_i using the algorithms of [11,37]. That is, the rebel individuals of EDA-MEC are obtained from $x_i = F_i^{-1}(1 - u_i)$. It must be highlighted that the rebels do not necessarily have to be obtained from the complement of samples; they can be generated from any process able to explore other regions of the search space.

Another proposed approach for preserving diversity in the EDA-MEC occurs in the sixth module and is a heuristic that reinitializes the population whenever:

- The linear correlation matrix is not positive definite (i.e., in instances when some or all eigenvalues are negative);
- Some variable assumes the same value in all individuals, which makes the linear correlation matrix generate invalid numeric values (NaN – Not a Number); or
- The norm of the difference between the maximum and minimum values of the individuals in each normalized variable with respect to the Euclidean distance is less than 10^{-4}. When this occurs, individuals are very close to each other and likely are stagnated.

Moreover, the evolutionary process of the EDA-MEC is elitist, always retaining the best individual of each generation. Whenever at least one of the aforementioned conditions is met, a population's percentage is restarted uniformly:

(a) in the real interval of search space and
(b) within the limits of the best individual variables at that generation.

The initial percentage is 20% "a") and 80% "b"). When the EDA-MEC cannot find a better individual after two consecutive restarts, the percentage is changed by 10%; i.e., "a") decreases to 10%, and "b") increases to 90%. In the next event, it is 0% "a") and 100% "b"). Subsequently, it is 90% "a") and 10% "b"), and so on.

4.2. EDA-MEC pseudocode

Considering what has been explained so far, the mechanism of EDA-MEC is stated as the pseudocode shown in Algorithm 1.

In steps 2 and 3, the experiment's initial settings are specified: the benchmark function (Φ) and its number of variables (n), the stop criteria, the copula function, the type of marginal distribution (F_n^{-1}), the number of individuals (m), the number of new individuals sampled from the copula (y), and the number of rebels (r).

In step 5, the EDA-MEC initializes a random population within the search space of the problem. This population is then evaluated, and the correlation matrix for all variables is calculated using the current population as the data inputs in steps 6 and 7. In the next step, if a premature convergence condition is identified (as defined in Section 4.1), the best individual is retained, a population's percentage is restarted uniformly in the real interval of search space and the remaining population is restarted uniformly within the limits of the best individual variables at that generation. Subsequently, a new correlation matrix is calculated.

In step 13, the EDA-MEC y samples are drawn from the copula using an algorithm for the simulation of random variables with copulas. Using these samples, in step 14 y new individuals are generated. After a transformation in the samples generated in step 13, r rebels are generated in step 15. In step 16, this new population is merged with the older. A new evaluation is performed, and the worst individuals are eliminated. Finally, the r individuals obtained from a variation of the learned CDF and having a lower probability of emerging in the evolutionary process (rebel individuals) replace the worst individuals from the resulting population. This entire process is repeated until a stop condition is attained, and the best individual is found.

Algorithm 1 Pseudocode for EDA-MEC.

```
Begin
  Definitions: benchmark (Φ), number of variables (n), number of individuals (m), number of new individuals (y) and number of rebels (r), and
   stop conditions;
  Choose a copula: Gaussian or Student's t; and the type of marginal distributions: normal or empirical (F_n^{-1});
   t → 0;
  Generate an initial population P(t) randomly in the search space: matrix (m individuals, n variables);
  Evaluate Φ(P(t));
  while (termination criteria are not satisfied) do
    Compute the linear correlation matrix R using P(t) as data inputs: (n, n);
    if (any of item 4.1 restart conditions is met) then
      Restart population P(t) using item 4.1 heuristic, retaining only the best individual;
      Compute the new linear correlation matrix R using P(t) as data inputs: (n, n);
    End
    Generate u_{y,n} samples using the algorithms for simulation of random variables with copulas;
    Generate new individuals using the samples of previous step in x_{y.n} = F_n^{-1}(u_{y,n});
    Generate rebel individuals: calculate 1 − u_{y,n}, select the top r (u_{r,n}), and use them in x_{r,n}^{rebel} = F_n^{-1}(u_{r,n});
    Merge the populations: x_{y,n} and P(t). Evaluate, retain the top m, and eliminate the remaining individuals;
    Replace the r worst of P(t) with x_{r,n}^{rebel};
     t → t + 1;
   End
  Output: the best individual
End
```

The proposed EDA-MEC algorithm has been evaluated and compared with other global optimization models. The following section presents and discusses these case studies.

5. Case studies

There have been many test or benchmark functions for global optimization problems reported in the literature; however, there is no standard list or set of benchmark functions. Ideally, test functions should have diverse properties such as separability, multimodality, and non-convexity, so that they can be truly useful for testing new algorithms in an unbiased way.

In the case of copula-based EDAs, there is a shortage of publicly available implementations and performance results for such algorithms. Recently, [19,20] implemented five models of multivariate CEDAs in the copulaedas package for the R language and performed benchmark tests. Thus, the same six benchmark functions and their search spaces used in [19] were also used in the evaluation of EDA-MEC, defined as the following:

- $f_{\text{Sphere}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$

  $\text{search space} = [-600,\ 600]$

  $\text{global minimum at } \boldsymbol{x} = (0, \ldots,\ 0)$ (11)

- $f_{\text{Ackley}}(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + \exp(1)$

  $\text{search space} = [-30,\ 30]$

  $\text{global minimumat } \boldsymbol{x} = (0, \ldots,\ 0)$ (12)

- $f_{\text{Rastrigin}}(\boldsymbol{x}) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$

  $\text{search space} = [-5.12,\ 5.12]$

  $\text{global minimum at } \boldsymbol{x} = (0, \ldots,\ 0)$ (13)

- $f_{\text{Griewank}}(\boldsymbol{x}) = 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$

  $\text{search space} = [-600,\ 600]$

  $\text{global minimum at } \boldsymbol{x} = (0, \ldots,\ 0)$ (14)

- $f_{\text{Summation Cancellation}}(\boldsymbol{x}) = \dfrac{-1}{10^{-5} + \sum_{i=1}^{n} |y_i|},\ y_1 = x_1,\ y_i = y_{i-1} + x_i$
  search space $= [-0.16,\ 0.16]$
  global minimum at $\boldsymbol{x} = (0, \ldots,\ 0)$ (15)

- $f_{\text{Rosenbrock}}(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(100\left(x_i^2 - x_{1+i}\right)^2 + (x_i - 1)^2\right)$
  search space $= [-9,\ 11]$
  global minimum at $\boldsymbol{x} = (1, \ldots,\ 1)$ (16)

In the interest of comparing the EDA-MEC performance results with those of [19], we adopted a similar experimental setup:

- All tests were performed with $n = 10$ variables to be optimized.
- There were 30 independent runs for each benchmark function, and the results are averaged over these independent trials. At the end of every trial, the EDA-MEC presents the following results: the best and worst fitness values found; the success rate; the MBF (Mean Best Fitness) of all runs and its standard deviation; the AES (Average number of Evaluations to Solution) of successful runs and its standard deviation; and the average CPU running time and its standard deviation.
- Two stop criteria were used: (i) reaching the global optimum with a precision greater than 10^{-6} or (ii) reaching 300,000 evaluations. They differ from the criteria defined for the IEEE Congress on Evolutionary Computation (CEC) 2005 Special Session on Real-Parameter Optimization, despite the fact that the benchmark functions are similar [56].

Because the models of [19] do not restart the population as the EDA-MEC does, they use a third stop criterion: the loss of diversity, namely, whenever the standard deviation of the evaluation of the solutions in the population is less than 10^{-8}. Incidentally, a successful variant of the non-elitist CMA-ES also uses a restart mechanism combined with an increasing population size that outperforms the pure restart approach on many multimodal problems [2]. Unlike the CMA-ES, the EDA-MEC does not increase the number of individuals after restarts and never decreases its current fitness level given that the evolution is elitist.

A key parameter in evolutionary algorithms is the population size. Hauschild and Pelikan [25] argues that it cannot be so small as to hinder the exploitation of the search space, nor too big, which increases the complexity of building and sampling probabilistic models. Thus, the population size has a strong influence on the performance of the evolutionary search. CMA-ES, for example, works well with relatively small population sizes. On the other hand, in [19] the bisection method is used to determine empirically the minimum population size required by the algorithm to find the global optimum of the function with a high success rate. This feature, although valuable for practical applications, hampers the comparison between algorithms and the reproducibility of results in experimental contexts.

In particular, we explored the following combination of parameter values:

- Marginal distributions: empirical and normal;
- Population size: 15, 20, and 30 individuals;
- Number of rebels: 0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 12, and 18 individuals; and
- Copulas: Gaussian and Student's t.

The results presented in the next subsections were derived from a statistical analysis that aimed to find the optimal parameters, i.e., those capable of producing the maximum number of convergences to the global optimum in each benchmark, with the lowest number of evaluations and machine time. All the experiments were executed on a machine with an Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz

5.1. Experiments with marginal distributions

The influence of the choice of marginal distributions on the EDA-MEC performance was verified with the Gaussian and Student's t copulas by the Average number of Evaluations to Solution (AES) and the success rate, as shown in Figs. 4 and 5, respectively.

It is found that in almost all benchmarks, except in the Rastrigin and Rosenbrock problems, the EDA-MEC performs better, on average, when both the Gaussian and Student's t copulas are constructed using normal marginal distributions. This can be explained by the simplicity of interactions between variables, in which modeling by a normal distribution is a cost-effective approach.

As can be seen in Fig. 5, the EDA-MEC is particularly efficient in terms of convergence to the global optimum of the Sphere, Ackley, Griewank and Summation Cancellation benchmarks when either copula is used along with the normal marginal. However, it is noted that the model fails in all tests of the Summation Cancellation when combining either of the copulas with the empirical marginal. Two hypotheses could explain this:

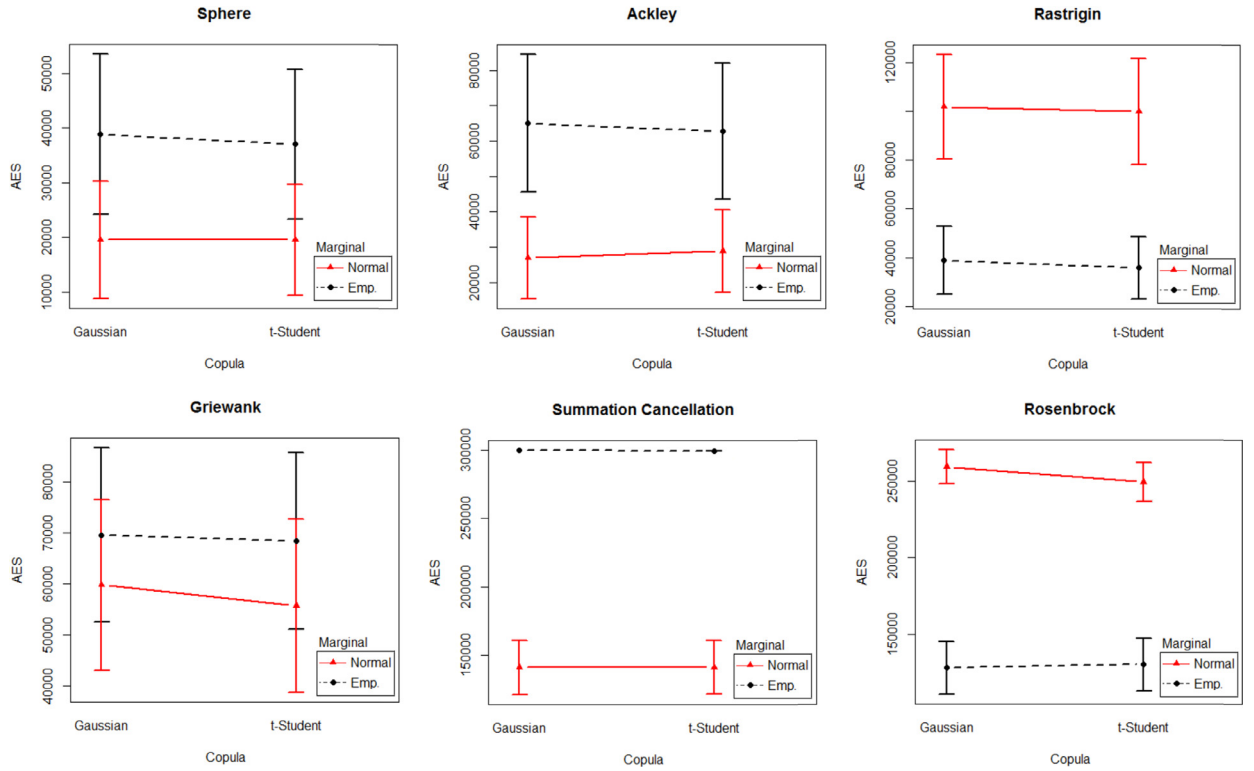

Fig. 4. Average number of evaluations to solution (AES) of the EDA-MEC, considering the interaction between two types of copulas and two marginal distributions.

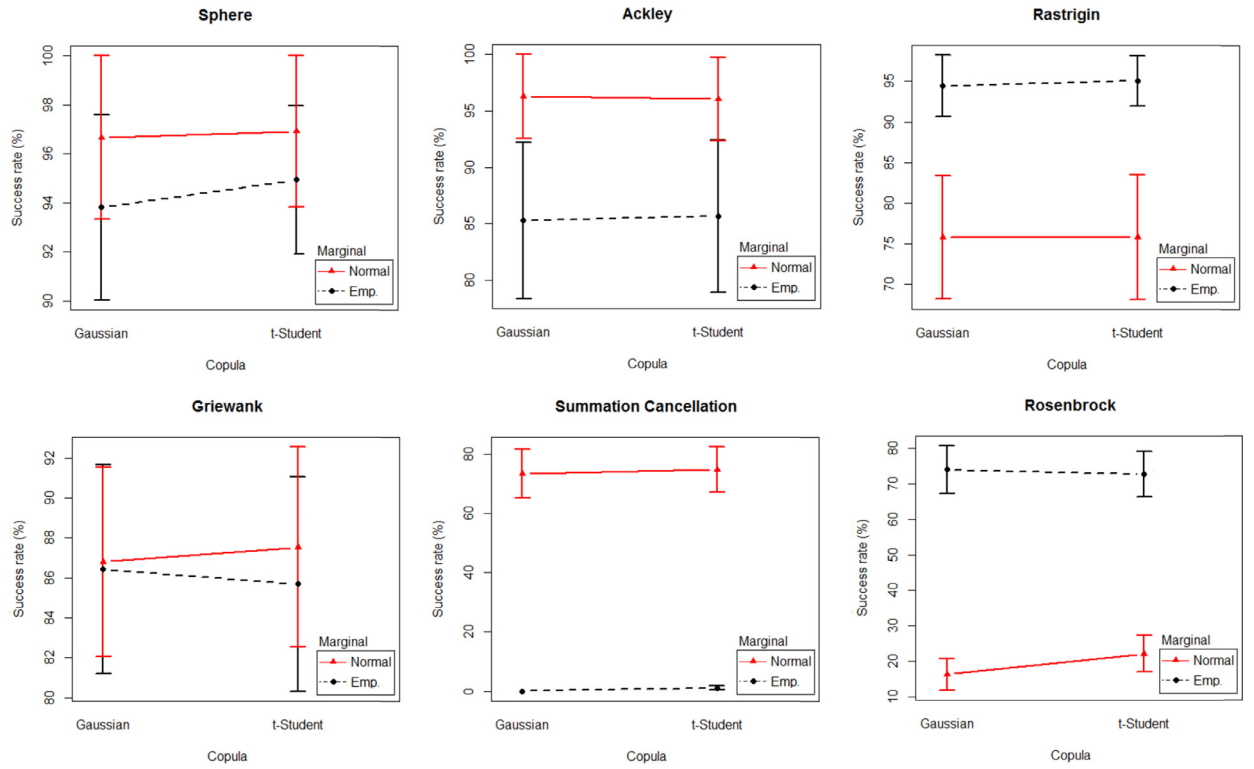

Fig. 5. Average success rate of the EDA-MEC, considering the interaction between two types of copulas and two marginal distributions.

- The normal distribution has a greater spatial density than the empirical, covering the entire search space. During the evolutionary process, the empirical distribution – based on the population of that generation – ceases to represent certain regions of the search space where the probability is zero (no individuals), and therefore the optimization fails.
- When using the empirical distribution, the EDA-MEC reassigns values from previous iterations (representing local optima) for at least one variable. In this case, it is not the fact that the empirical marginal is combined with a copula that makes the algorithm work well, but the features of the benchmark function. Thus, the performance should be evaluated by combining the benchmark function and the marginal distribution. This can also explain the poor EDA-MEC performance on optimizing Rosenbrock using the normal marginal joint with any of the copulas. However, we cannot dismiss the influence of the copula chosen for optimization. For example, the choice of a tail-dependent copula that generates random numbers from the extreme value distribution may not be suitable for certain problems.

5.2. Experiments with different population sizes

The EDA-MEC performance in terms of evaluations, machine time, and success rate cannot be measured exclusively from the number of individuals defined at the beginning of each test; another important factor has an influence: the number of individuals restarting. To illustrate this interaction, the following graphs show the relationship between the type of copula and the population size, by the average of:

- Average number of Evaluations to Solution (AES);
- CPU running time;
- Success rate; and
- Number of individuals restarting.

The curves of Fig. 6 display the regular behavior of the EDA-MEC with both copulas in most benchmarks: smaller populations cause a greater number of restarts and consequently a longer average machine time, which does not result in a proportional increase in the Average number of Evaluations to Solution (AES). The EDA-MEC with 15 individuals restarts 3 and 8 times more than with 30 individuals in the Sphere and Rosenbrock benchmarks, respectively, thereby demanding more machine time. However, a proportional increase in the AES is not observed: in the Sphere, the number of restarts equals the total number of individuals, which results in a very close AES for the two scenarios; in Rosenbrock, the EDA-MEC requires fewer evaluations for the smallest population, as in this case, and it can converge more quickly to the global optimum. Although the higher number of restarts does not necessarily contribute to outperforming trials with larger populations in terms of the average success rate, these restarts play an important role in correcting premature convergence, allowing the EDA-MEC to achieve good results.

It is important to emphasize the importance of restarts in EDA-MEC. Trials not presented in this paper have shown that the algorithm stagnates when the function that promotes restarts is disabled. Complementary to the circumstances described in Section 4.1, it was found that restarts occurred whenever individuals were only slightly scattered, that is, when the population's standard deviation converged to zero. However, there has been no regular behavior of the mean and standard deviation of the correlation matrix that would allow a generalization of an implicit condition for restarting.

5.3. Experiments with different numbers of rebel individuals

The inclusion of rebels in the population, an innovation contained in the EDA-MEC, together with the heuristic to reinitialize the population, helps this copula-based EDA to deal with the problem of premature convergence. The following figure illustrates the impact of the rebels on the EDA-MEC performance in optimizing each of the six benchmark functions in trials with populations of 15, 20 and 30 individuals; with the Gaussian and Student's t copulas; and with the empirical and normal marginal distributions. Thus, the standard deviations in Fig. 7 involve the use of different populations, types of copulas and marginal distributions.

It can be observed from Fig. 7 that the EDA-MEC does not need rebels to optimize the simplest functions, Sphere and Ackley, in all experiments. Furthermore, the EDA-MEC does require rebels to increase the success rate in all other benchmark functions. Particularly in the most difficult one, the Rosenbrock, it is observed that:

- The best median occurs when the number of rebels is equal to one; and the worst at zero rebels;
- The worst case with a rebel in the population is better than all other worst cases, that is, the lower limit of the case with a rebel is superior to lower limits with other rebel settings;
- With a rebel, the results are less dispersed, concentrated between the 2nd and 3rd quartiles; and
- Not using rebels provides the worst median result.

Thus, the use of rebels is critical for improving the performance of EDA-MEC.

5.4. The comparison with other copula-based EDAs and CMA-ES

After carrying out the previous exploratory experiments it is essential to carry out a comparative study that assesses EDA-MEC performance with regard to comparable state of the art approaches. This section presents the results of that

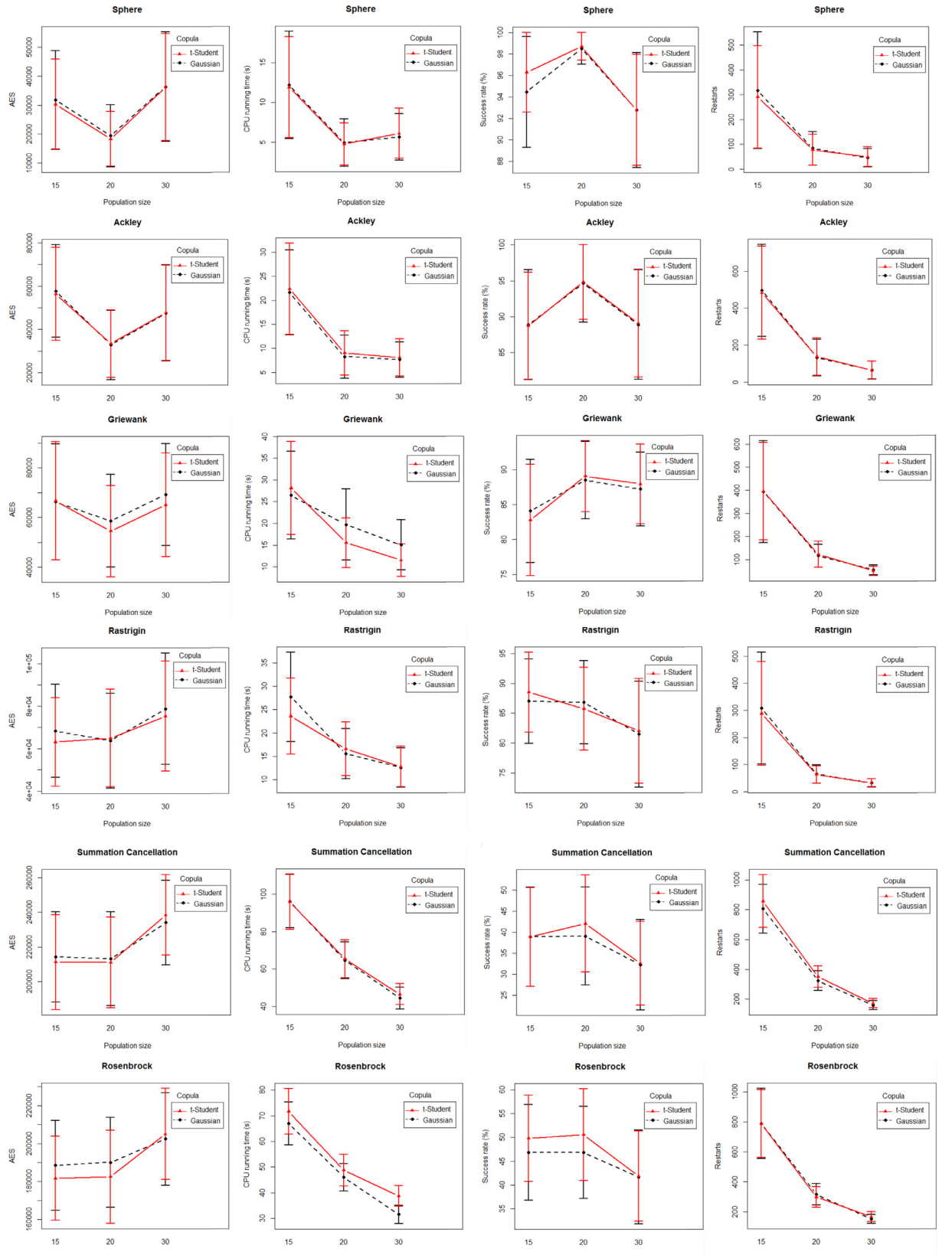

Fig. 6. Behavior of four performance measures of the EDA-MEC with different copulas and population sizes in Sphere, Ackley, Griewank, Rastrigin, Summation cancellation, and Rosenbrock benchmarks.

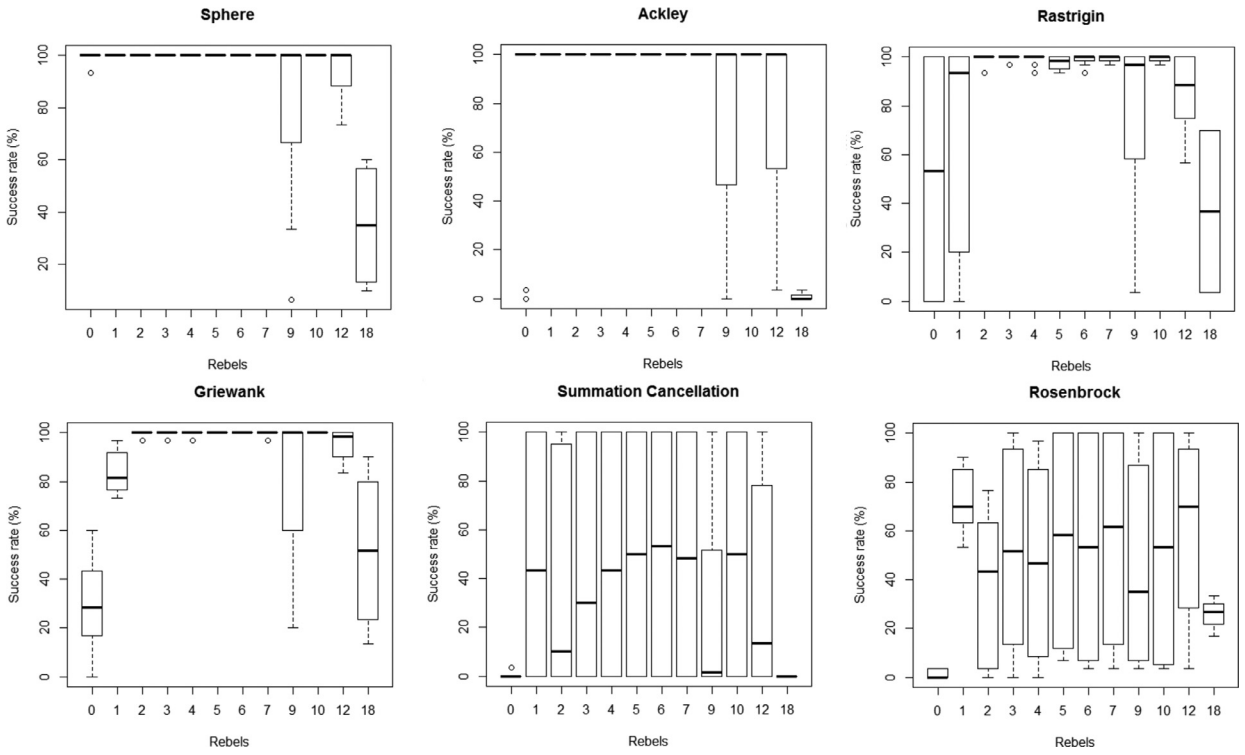

Fig. 7. Influence of the rebels on the convergence of the EDA-MEC in the six benchmark functions.

Table 1
Sphere problem.

Algorithm	Pop. size	Success rate	Best fitness	Worst fitness	MBF	AES	CPU running time (s)
EDA-MEC*	**20**	**30/30**	$\mathbf{1.6 \times 10^{-7}}$	$\mathbf{9.9 \times 10^{-7}}$	$\mathbf{6.1 \times 10^{-7} \pm 2.6 \times 10^{-7}}$	$\mathbf{486 \pm 25}$	$\mathbf{0.1 \pm 0.02}$
EDA-MEC**	**20**	**30/30**	$\mathbf{2.7 \times 10^{-7}}$	$\mathbf{9.9 \times 10^{-7}}$	$\mathbf{5.7 \times 10^{-7} \pm 2.0 \times 10^{-7}}$	$\mathbf{568 \pm 354}$	$\mathbf{0.1 \pm 0.06}$
UMDA	81	30/30	–	–	$6.9 \times 10^{-7} \pm 2.3 \times 10^{-7}$	3823 ± 128.3	0.4 ± 0.0
GCEDA	310	30/30	–	–	$6.5 \times 10^{-7} \pm 2.0 \times 10^{-7}$	$13{,}082 \pm 221.4$	0.9 ± 0.0
CVEDA	104	30/30	–	–	$6.7 \times 10^{-7} \pm 1.8 \times 10^{-7}$	4777 ± 118.8	8.3 ± 1.5
DVEDA	104	30/30	–	–	$6.7 \times 10^{-7} \pm 2.0 \times 10^{-7}$	4787 ± 100.2	8.2 ± 1.1
Copula MIMIC	150	30/30	–	–	$6.9 \times 10^{-7} \pm 1.7 \times 10^{-7}$	6495 ± 209.0	468.8 ± 62.9
CMA-ES	10	30/30	2.9×10^{-7}	9.9×10^{-7}	$7.7 \times 10^{-7} \pm 1.7 \times 10^{-7}$	2119 ± 96	0.29 ± 0.02

* Copula: Student's t; Marginal: normal; numbers of rebels = 7; average number of restarts: 0 ± 0.
** Copula: Gaussian; Marginal: normal; numbers of rebels = 10; average number of restarts: 0.1 ± 0.55.

comparative study. In particular, we compared EDA-MEC with the best results obtained with the [19,20] models and CMA-ES. Only linear and independence relationships were considered in the models of [19]. Thus, only normal and product copulas were used in these EDAs.

As part of the above debated experiments, various trials were performed to evaluate the impact of different parameters on EDA-MEC's performance, which allowed the identification of the parameter combination across the different problems, which was used to in this experiment. The EDA-MEC model was based on both the Gaussian and Student's t copulas and used different marginal distributions, population sizes and numbers of rebels in order to get the best result for each benchmark function.

In the case of CMA-ES, we used the source code (version 3.61.beta) available on the website of the author of the algorithm. [1] In order to obtain a fair comparison with EDA-MEC and CMA-ES, similar configurations were adopted, with various stop criteria that the CMA-ES code provides being disabled. Even so, hidden restrictions hard-coded in the CMA-ES implementation stopped the execution before the 300,000 evaluations, when the CMA-ES stagnated. This means that in trials where the method failed, that is, when the success rate was not 30/30, the number of evaluations is certainly higher than reported in Tables 1 to 6.

[1] https://www.lri.fr/~hansen/cmaes_inmatlab.html#matlab

Table 2
Ackley problem.

Algorithm	Pop. size	Success rate	Best fitness	Worst fitness	MBF	AES	CPU running time (s)
EDA-MEC*	**20**	**30/30**	$\mathbf{5.3 \times 10^{-7}}$	$\mathbf{9.9 \times 10^{-7}}$	$\mathbf{8.1 \times 10^{-7} \pm 1.3 \times 10^{-7}}$	**638 ± 34**	**0.12 + 0.01**
EDA-MEC**	**20**	**30/30**	$\mathbf{5.5 \times 10^{-7}}$	$\mathbf{9.6 \times 10^{-7}}$	$\mathbf{7.8 \times 10^{-} \pm 1.2 \times 10^{-7}}$	**857 ± 745**	**0.16 ± 0.13**
UMDA	81	30/30	–	–	$8.1 \times 10^{-7} \pm 1.1 \times 10^{-7}$	4998 88.0	0.6 ± 0.0
GCEDA	279	30/30	–	–	$8.3 \times 10^{-7} \pm 1.1 \times 10^{-7}$	15,633 ± 258.8	1.6 ± 0.0
CVEDA	104	30/30	–	–	$8.1 \times 10^{-7} \pm 1.0 \times 10^{-7}$	6330 ± 163.2	10.6 ± 1.8
DVEDA	111	30/30	–	–	$7.9 \times 10^{-7} \pm 1.4 \times 10^{-7}$	6679 ± 133.8	11.4 ± 1.6
Copula MIMIC	188	30/30	–	–	$8.0 \times 10^{-7} \pm 1.2 \times 10^{-7}$	10,784 ± 143.7	800.1 ± 122.1
CMA-ES	10	29/30	8.7×10^{-7}	$1.9 \times 10^{+1}$	$1.9 \times 10^{+11} \pm 3.5 \times 10^{0}$	25,245 ± 2.595	1.02 ± 0.02

* Copula: Student's t; Marginal: normal; numbers of rebels = 10; average number of restarts: 0 ± 0.
** Copula: Gaussian; Marginal: normal; numbers of rebels = 7; average number of restarts: 0.1 + 0.4.

Table 3
Rastrigin problem.

Algorithm	Pop. size	Success rate	Best fitness	Worst fitness	MBF	AES	CPU running time (s)
EDA-MEC*	**30**	**30/30**	$\mathbf{2.5 \times 10^{-7}}$	$\mathbf{9.8 \times 10^{-7}}$	$\mathbf{7.0 \times 10^{-7} \pm 2.0 \times 10^{-7}}$	**6141 ± 2938**	**1.43 + 0.67**
EDA-MEC**	**20**	**30/30**	$\mathbf{7.4 \times 10^{-7}}$	$\mathbf{9.7 \times 10^{-7}}$	$\mathbf{6.6 \times 10^{-7} \pm 2.7 \times 10^{-7}}$	**7076 ± 3125**	**29.07 + 15.25**
UMDA	447	30/30	–	–	$6.7 \times 10^{-7} \pm 2.3 \times 10^{-7}$	33,614 ± 2452	1.7 ± 0.1
GCEDA	721	30/30	–	–	$6.8 \times 10^{-7} \pm 1.8 \times 10^{-7}$	46,095 ± 2158	2.8 ± 0.1
CVEDA	447	30/30	–	–	$6.6 \times 10^{-7} \pm 1.7 \times 10^{-7}$	32,914 ± 2011	44.8 ± 13.3
DVEDA	325	30/30	–	–	$7.3 \times 10^{-7} \pm 1.7 \times 10^{-7}$	24,710 ± 1754	31.3 ± 6.9
Copula MIMIC	386	30/30	–	–	$6.4 \times 10^{-7} \pm 2.1 \times 10^{-7}$	27,315 ± 1673.9	1.432 ± 218.6
CMA-ES	10	27/30	5.6×10^{-7}	$1.3 \times 10^{+2}$	$4.4 \times 10^{+1} \pm 3.9 \times 10^{+1}$	226,56 ± 93,76	7.66 ± 7.3

* Copula: Gaussian; Marginal: empirical; numbers of rebels = 12; average number of restarts: 11.1 ± 7.4.
** Copula: Student's t; Marginal: empirical; numbers of rebels = 7; average number of restarts: 29.1 + 15.3.

Table 4
Griewank problem.

Algorithm	Pop. size	Success rate	Best fitness	Worst fitness	MBF	AES	CPU running time (s)
EDA-MEC*	**15**	**30/30**	$\mathbf{3.1 \times 10^{-7}}$	$\mathbf{9.6 \times 10^{-7}}$	$\mathbf{6.55 \times 10^{-7} \pm 1.9 \times 10^{-7}}$	**2416 ± 2733**	**0.69 ± 0.79**
EDA-MEC**	**15**	**30/30**	$\mathbf{1.8 \times 10^{-7}}$	$\mathbf{9.9 \times 10^{-7}}$	$\mathbf{7.19 \times 10^{-7} \pm 1.96 \times 10^{-7}}$	**3147 ± 3245**	**0.84 ± 0.86**
UMDA	111	30/30	–	–	$6.6 \times 10^{-7} \pm 1.9 \times 10^{-7}$	5224 ± 231,2	0.5 ± 0.0
GCEDA	355	30/30	–	–	$6.9 \times 10^{-7} \pm 1.8 \times 10^{-7}$	15,099 ± 414,1	1.2 ± 0.0
CVEDA	142	30/30	–	–	$7.0 \times 10^{-7} \pm 1.8 \times 10^{-7}$	6579 ± 389,9	9.3 ± 2.0
DVEDA	150	30/30	–	–	$6.5 \times 10^{-7} \pm 2.4 \times 10^{-7}$	6785 ± 338,1	9.5 ± 1.9
Copula MIMIC	188	30/30	–	–	$6.6 \times 10^{-7} \pm 1.8 \times 10^{-7}$	8221 ± 220,4	595.7 ± 91.6
CMA-ES	10	30/30	2.9×10^{-7}	9.9×10^{-7}	$7.6 \times 10^{-7} \pm 1.7 \times 10^{-7}$	76,099 ± 83,741	0.49 ± 0.19

* Copula: Student's t; Marginal: normal; numbers of rebels = 6; average number of restarts: 2.1 ± 2.8.
** Copula: Gaussian; Marginal: normal; numbers of rebels = 4; average number of restarts: 2.1 + 2.2.

Table 5
Summation cancellation problem.

Algorithm	Pop. size	Success rate	Best fitness	Worst fitness	MBF	AES	CPU running time (s)
EDA-MEC*	**30**	**30/30**	$\mathbf{-1 \times 10^{+5}}$	$\mathbf{-1 \times 10^{+5}}$	$\mathbf{-1 \times 10^{+5} \pm 1.2 \times 10^{-7}}$	**13,802 ± 7569**	**1.7 ± 0.95**
EDA-MEC**	**20**	**30/30**	$\mathbf{-1 \times 10^{+5}}$	$\mathbf{-1 \times 10^{+5}}$	$\mathbf{-1 \times 10^{+5} \pm 2.1 \times 10^{-7}}$	**14,442 ± 8698**	**2.8 ± 1.7**
UMDA	2000	0/30	–	–	$-5.7E \times 10^{+2} \pm 3.4 \times 10^{+2}$	300,000 ± 0.0	66.8 ± 0.7
GCEDA	355	30/30	–	–	$-1 \times 10^{+5} \pm 1.3 \times 10^{-7}$	42,434 ± 305	9.3 ± 0.4
CVEDA	325	30/30	–	–	$-1 \times 10^{+5} \pm 1.3 \times 10^{-7}$	44,622 ± 858	537.2 ± 6.6
DVEDA	965	30/30	–	–	$-1 \times 10^{+5} \pm 9.3 \times 10^{-8}$	117,408 ± 959	2.367 ± 20
Copula MIMIC	2000	30/30	–	–	$-2.3 \times 10^{+4} \pm 2.7 \times 10^{+4}$	300,000 ± 0.0	10.426 ± 1054
CMA-ES	10	30/30	$-1 \times 10^{+5}$	$-1 \times 10^{+5}$	$-1 \times 10^{+5} \pm 7.0 \times 10^{-8}$	7483 ± 5.004	1.0 ± 0.67

* Copula: Gaussian; Marginal: normal; numbers of rebels = 12; average number of restarts: 1.8 ± 1.5.
** Copula: Student's t; Marginal: normal; numbers of rebels = 10; average number of restarts: 7.3 + 5.2.

Tables 1 to 6 present the comparative results for each benchmark function. Below each table, the input parameters (copula, the marginal distribution, and the numbers of rebels) and the output (the numbers of restarts occurring during the evolution) are specified.

In the following comparative tables, the model presented in this paper (EDA-MEC) is highlighted in bold, and the method that had the best performance in that benchmark function is shaded. The criteria for best performance are considered in

Table 6
Rosenbrock problem.

Algorithm	Pop. size	Success rate	Best fitness	Worst fitness	MBF	AES	CPU running time (s)
EDA-MEC*	**30**	**30/30**	$\mathbf{8.1 \times 10^{-11}}$	$\mathbf{9.9 \times 10^{-7}}$	$\mathbf{5.7 \times 10^{-7} \pm 3.1 \times 10^{-7}}$	**25,137 ± 19,597**	**5.7 ± 4.6**
EDA-MEC**	**20**	**30/30**	$\mathbf{8.8 \times 10^{-9}}$	$\mathbf{9.7 \times 10^{-7}}$	$\mathbf{5.5 \times 10^{-7} \pm 3.1 \times 10^{-7}}$	**35,546 ± 26,495**	**10.2 ± 8.2**
UMDA	2000	0/30	–	–	$8.0 \times 10^{+0} \pm 2.6 \times 10^{-2}$	300,000 ± 0,0	72.0 ± 0.1
GCEDA	2000	0/30	–	–	$7.5 \times 10^{-0} \pm 1.9 \times 10^{-1}$	300,000 ± 0,0	74.3 ± 0.4
CVEDA	2000	0/30	–	–	$7.5 \times 10^{+0} \pm 1.1 \times 10^{-1}$	193,867 ± 48,243	1.279 ± 532
DVEDA	2000	0/30	–	–	$7.5 \times 10^{+0} \pm 1.5 \times 10^{-1}$	172,200 ± 35,184	961 ± 386
Copula MIMIC	2000	0/30	–	–	$7.6 \times 10^{+0} \pm 1.3 \times 10^{-1}$	139,000 ± 5139	6.174 ± 821
CMA-ES	10	30/30	8.2×10^{-7}	1.0×10^{-6}	$8.2 \times 10^{-7} \pm 1.6 \times 10^{-7}$	6212 ± 1103	0.89 ± 0.16

* Copula: Student's t; Marginal: empirical; numbers of rebels = 12; average number of restarts: 70.9 ± 51.0.
** Copula: Gaussian; Marginal: empirical; numbers of rebels = 10; average number of restarts: 159.7 + 73.0.

this order: (i) the best success rate, (ii) the lowest AES (Average number of Evaluations to Solution), and (iii) the best MBF (Mean Best Fitness).

It is possible to verify that the proposed algorithm provides superior results for most functions, thus yielding a smaller number of evaluations to achieve the optimization target. It had the best performance in the Summation Cancellation and Rosenbrock, for which the CMA-ES achieved the target with one-half and one-fourth, respectively, of the number of EDA-MEC evaluations.

In Sphere, Ackley and Griewank, in which there are no strong dependencies between the variables of these functions, all methods reached the global minimum in all experiments except the CMA-ES, which failed to optimize Ackley in any experiment. Importantly, a few parameters need to be set by the CMA-ES user: the benchmark function and its number of variables, and the termination criteria. Even the population size is a function of the number of variables and is comparatively small to allow for fast convergence. An optional restart parameter with increasing population size to improve the global search performance was used in all trials for all benchmarks. It was observed that without it, the CMA-ES failed to optimize Rosenbrock in some trials but converged anyway with this parameter. Nevertheless, this configuration did not work well on the Ackley nor on the Rastrigin, for which the EDA-MEC failed in three experiments.

Regarding the algorithms of [19], the UMDA had the best performance in Sphere, Ackley and Griewank; the others need to calculate a larger number of parameters to represent the relationships between the variables. Therefore, larger populations were needed to calculate them reliably, according to the authors.

In Rastrigin, which is not a function where the interactions between variables play an important role in the success of optimization, all models reached the global optimum except the CMA-ES. In this benchmark, [19] observed that the combination of normal and product copulas in a single probabilistic model is better than the assumption of independence or a multivariate linear dependence structure, justifying the best performance of DVEDA and Copula MIMIC over the other algorithms of those authors.

In Summation Cancellation, a correct representation of the strong interactions between variables is essential for the algorithm to optimize it. Thus, these algorithms fail: UMDA, which ignores the need to assume independence among variables, and the Copula MIMIC, the model that cannot represent the dependencies for successful optimization.

Finally, Rosenbrock is recognized as a difficult problem for most optimization algorithms with nonlinear dependence between variables; only the EDA-MEC and the CMA-ES were able to optimize it.

5.5. Other experiments

Table 7 exhibits the EDA-MEC performance when the degrees of freedom (the Student's t copula) are obtained by MLE and results from the previous sections for the six benchmarks with 10 variables. In the first two rows of each benchmark are observed very similar results between the Gaussian and Student's t copulas, although the number of degrees of freedom fixed and with a low value has been deliberately chosen so that the t distribution does not converge to the Gaussian. This performance similarity can be explained by a common characteristic of these benchmarks: a global optimum in the center of the domain. Note that the greatest difference in performance occurred in Rosenbrock, which has no central global optimum in an asymmetric domain. With regard to the use of MLE in the calculation of degrees of freedom for each generation of EDA-MEC, there is a reduction in AES, except in the Sphere and Summation Cancellation. However, this gain is not worthwhile given the considerable increase that occurred in the computational time. Thus the methodology adopted in the trials with the benchmarks of this article is validated: the use of a fixed and small number of degrees of freedom in the Student's t copula.

The majority of evolutionary algorithms lose the power of searching the optimal solution when the dimensionality of the problem increases. Nevertheless, Table 8 shows that EDA-MEC remains a competent optimization algorithm for solving problems with up to 100 dimensions. The EDA-MEC performance was evaluated using the previous benchmark functions with 10, 50, and 100 variables. After systematically testing a range of population sizes (from 110 to 400 individuals) and a number of rebels approximately equal to 25% of the population size, it was found the smallest population size capable of optimizing most problems. All tests were performed with 300 individuals and 70 rebels and using the Gaussian copula and

Table 7
The performance comparison of EDA-MEC using elliptical copulas with fixed degrees of freedom (D.F.) and estimated by MLE in benchmark functions with 10 variables.

Benchmark	Copula	D.F.	Size pop	Num rebels	Success rate	Average number of restarts	MBF	AES	CPU running time (s)
Sphere	Gaussian	–	20	10	30/30	$0{,}1 \pm 0{,}6$	$5{,}7 \times 10^{-7} \pm 2{,}0 \times 10^{-7}$	568 ± 354	$0{,}10 \pm 0{,}06$
	Student's t	10	20	10	30/30	$0{,}1 \pm 0{,}3$	$6{,}7 \times 10^{-7} \pm 2{,}2 \times 10^{-7}$	506 ± 105	$0{,}11 \pm 0{,}03$
	Student's t	MLE	20	10	30/30	$0{,}3 \pm 0{,}7$	$5{,}9 \times 10^{-7} \pm 2{,}8 \times 10^{-7}$	525 ± 308	$12{,}03 \pm 7{,}06$
Ackley	Gaussian	–	30	12	30/30	0 ± 0	$8{,}0 \times 10^{-7} \pm 1{,}5 \times 10^{-7}$	1.051 ± 50	$0{,}13 \pm 0{,}01$
	Student's t	15	30	12	30/30	0 ± 0	$8{,}0 \times 10^{-7} \pm 1{,}5 \times 10^{-7}$	1.004 ± 41	$0{,}13 \pm 0{,}01$
	Student's t	MLE	30	12	30/30	0 ± 0	$7{,}7 \times 10^{-7} \pm 1{,}5 \times 10^{-7}$	877 ± 30	$22{,}9 \pm 1{,}34$
Rastrigin*	Gaussian	–	20	7	30/30	32 ± 17	$7{,}0 \times 10^{-7} \pm 2{,}5 \times 10^{-7}$	7.466 ± 3.499	$2{,}53 \pm 1{,}19$
	Student's t	10	20	7	30/30	29 ± 15	$6{,}6 \times 10^{-7} \pm 2{,}7 \times 10^{-7}$	7.076 ± 3.125	$2{,}09 \pm 0{,}93$
	Student's t	MLE	20	7	30/30	23 ± 11	$7{,}8 \times 10^{-7} \pm 2{,}3 \times 10^{-7}$	5.649 ± 1.962	$169{,}94 \pm 58{,}19$
Griewank	Gaussian	–	15	7	30/30	3 ± 3	$6{,}9 \times 10^{-7} \pm 2{,}3 \times 10^{-7}$	3.431 ± 3.547	$0{,}90 \pm 0{,}94$
	Student's t	7	15	7	30/30	3 ± 5	$6{,}8 \times 10^{-7} \pm 2{,}3 \times 10^{-7}$	3.211 ± 5.770	$0{,}91 \pm 1{,}66$
	Student's t	MLE	15	7	30/30	3 ± 4	$6{,}5 \times 10^{-7} \pm 2{,}4 \times 10^{-7}$	1.825 ± 2.959	$50{,}49 \pm 83{,}61$
Summation Cancellation	Gaussian	–	20	10	30/30	8 ± 6	$-1 \times 10^{+5} \pm 1{,}5 \times 10^{-7}$	14.336 ± 9.426	$2{,}68 \pm 1{,}79$
	Student's t	10	20	10	30/30	7 ± 5	$-1 \times 10^{+5} \pm 2{,}1 \times 10^{-7}$	14.442 ± 8.698	$2{,}82 \pm 1{,}72$
	Student's t	MLE	20	10	30/30	16 ± 9	$-1 \times 10^{+5} \pm 1{,}8 \times 10^{-7}$	20.072 ± 9.451	$460{,}77 \pm 213{,}45$
Rosenbrock*	Gaussian	–	15	5	30/30	360 ± 195	$5{,}8 \times 10^{-7} \pm 2{,}7 \times 10^{-7}$	40.219 ± 22.096	$15{,}04 \pm 8{,}42$
	Student's t	7	15	5	30/30	343 ± 198	$6{,}5 \times 10^{-7} \pm 2{,}5 \times 10^{-7}$	38.030 ± 23.151	$15{,}02 \pm 9{,}35$
	Student's t	MLE	15	5	30/30	332 ± 281	$6{,}3 \times 10^{-7} \pm 2{,}9 \times 10^{-7}$	35.549 ± 30.493	$827{,}60 \pm 721{,}69$

* Marginal: empirical; all other benchmarks used normal marginal distributions.

the Student's t copula with 1 and 15 °s of freedom (D.F.). As in many practical applications, the t distributions is actually replaced by the standard normal distribution when D.F. ≥ 30, the Student's t copula with 1 and 15 °s of freedom is clearly non-Normal.

Overall, the experimental results, over all dimensions, have shown that EDA-MEC using the Gaussian and the Student's *t* copula with 15 °s of freedom optimizes all benchmarks functions without exceeding the maximum number of 300,000 evaluations and with a non-linear growth of CPU running time. This result demonstrates the EDA-MEC is numerically stable and suggests it can optimize benchmark functions with more than 100 decision variables. Although the EDA-MEC using the Student's t copula with 1 ° of freedom have provided superior results, thus yielding a smaller CPU running time and a number of evaluations to achieve the optimization target for most functions, it failed to optimize Ackley and Rosenbrock problems with 10 variables (highlighted in the table). It also can be observed that, in these cases, the EDA-MEC model got stuck at a local optima and neither the rebels nor the population restarting (a great number) prevented premature convergence in a few runs. In these cases, the most prominent characteristic of the Student's *t* copula with 1 ° of freedom generated numbers from extreme value distribution that were not suitable for optimization.

6. Conclusion

This paper has presented a proposal for a new metaheuristic optimization model capable of identifying good solutions without large computational efforts, through efficient use of information about the search space, which was called Estimation of Distribution Algorithm based on Multivariate Elliptical Copulas (EDA-MEC).

The EDA-MEC uses techniques that are not commonly adopted in other Copula-based EDAs (CEDAs). One is the copula parameter updating; the other is related to the modeling of marginal distributions.

In the literature, most CEDAs use the Maximum Likelihood Estimator (MLE) and do not perform copula parameter updating. In EDA-MEC, the correlation matrix for all variables is calculated using the current population as the data inputs, which increases the reliability of dependency information among the variables during the algorithm's execution. Moreover, for the Student's t copula, fixing the degrees of freedom at a fairly. Low value, rather than calculating it by MLE, considerably reduced the computational cost of EDA-MEC in the benchmarks in this paper.

Another aspect that has not been explored adequately is in regard to marginal distributions. These algorithms do not check which type of marginal distribution is most appropriate in describing the behavior of the variables. By setting this important part of the model as a parameter, the EDA-MEC showed empirically that changing the marginal distribution type can determine the algorithm's convergence to the global optimum.

The EDA-MEC makes the contribution of a heuristic for restarting a population that, in addition to solving the problem of indeterminacy correlation between the variables, allows the EDA-MEC to insert the diversity needed to escape from local optima. This mechanism provides data to construct a new search distribution. A percentage of these data come from nearby variables of the best solution, and the rest are scattered randomly in the real interval of search space. Another diversity-preserving mechanism of EDA-MEC is its use of a variation of the concept of rebel individuals that are obtained by a modification of the original probabilistic model in order to explore other regions of the search space. Empirically, this

Table 8
The performance of EDA-MEC using elliptical copulas in benchmark functions with 10, 50, and 100 variables.

Benchmark	Num Variables	Copula	D.F.	Success rate	Avg num of restarts	MBF	AES	CPU running time (s)
Sphere	10	Gaussian	–	30/30	0 ± 0	$5.6 \times 10^{-7} \pm 2.7 \times 10^{-7}$	3155 ± 375	0.04 ± 0.01
		Student's t	15	30/30	0 ± 0	$4.9 \times 10^{-7} \pm 3.0 \times 10^{-7}$	3110 ± 297	0.05 ± 0.02
		Student's t	1	30/30	0 ± 0	$5.0 \times 10^{-7} \pm 2.8 \times 10^{-7}$	2930 ± 363	0.04 ± 0.01
	50	Gaussian	–	30/30	0 ± 0	$5.2 \times 10^{-7} \pm 2.6 \times 10^{-7}$	4190 ± 249	0.23 ± 0.03
		Student's t	15	30/30	0 ± 0	$5.1 \times 10^{-7} \pm 2.8 \times 10^{-7}$	3980 ± 218	0.27 ± 0.03
		Student's t	1	30/30	0 ± 0	$4.5 \times 10^{-7} \pm 2.5 \times 10^{-7}$	3155 ± 293	0.19 ± 0.02
	100	Gaussian	–	30/30	0 ± 0	$4.9 \times 10^{-7} \pm 2.5 \times 10^{-7}$	5000 ± 210	0.57 ± 0.03
		Student's t	15	30/30	0 ± 0	$5.4 \times 10^{-7} \pm 2.1 \times 10^{-7}$	4730 ± 251	0.67 ± 0.04
		Student's t	1	30/30	0 ± 0	$5.1 \times 10^{-7} \pm 2.7 \times 10^{-7}$	3400 ± 262	0.40 ± 0.04
Ackley	10	Gaussian	–	30/30	0 ± 0	$6.6 \times 10^{-7} \pm 2.3 \times 10^{-7}$	4120 ± 304	0.06 ± 0.01
		Student's t	15	30/30	0 ± 0	$6.5 \times 10^{-7} \pm 2.0 \times 10^{-7}$	4250 ± 458	0.08 ± 0.02
		Student's t	1	27/30	53.9 ± 155	$7.4 \times 10^{-5} \pm 3.9 \times 10^{-4}$	$36{,}775 \pm 90{,}917$	0.35 ± 0.82
	50	Gaussian	–	30/30	0 ± 0	$7.2 \times 10^{-7} \pm 2.2 \times 10^{-7}$	5160 ± 304	0.28 ± 0.03
		Student's t	15	30/30	0 ± 0	$6.6 \times 10^{-7} \pm 1.9 \times 10^{-7}$	4970 ± 254	0.36 ± 0.03
		Student's t	1	30/30	0 ± 0	$6.9 \times 10^{-7} \pm 2.3 \times 10^{-7}$	4025 ± 361	0.25 ± 0.03
	100	Gaussian	–	30/30	0 ± 0	$6.9 \times 10^{-7} \pm 2.0 \times 10^{-7}$	5985 ± 221	0.69 ± 0.04
		Student's t	15	30/30	0 ± 0	$6.8 \times 10^{-7} \pm 2.4 \times 10^{-7}$	5755 ± 241	0.84 ± 0.05
		Student's t	1	30/30	0 ± 0	$6.3 \times 10^{-7} \pm 2.2 \times 10^{-7}$	4235 ± 222	0.51 ± 0.04
Rastrigin	10	Gaussian	–	30/30	0 ± 0	$5.0 \times 10^{-7} \pm 2.6 \times 10^{-7}$	2710 ± 281	0.04 ± 0.01
		Student's t	15	30/30	0 ± 0	$4.9 \times 10^{-7} \pm 2.8 \times 10^{-7}$	2795 ± 335	0.05 ± 0.02
		Student's t	1	30/30	0.07 ± 0.37	$4.8 \times 10^{-7} \pm 2.9 \times 10^{-7}$	2985 ± 2638	0.04 ± 0.04
	50	Gaussian	–	30/30	0 ± 0	$5.9 \times 10^{-7} \pm 3.1 \times 10^{-7}$	3695 ± 195	0.20 ± 0.02
		Student's t	15	30/30	0 ± 0	$5.1 \times 10^{-7} \pm 2.6 \times 10^{-7}$	3490 ± 239	0.25 ± 0.02
		Student's t	1	30/30	0 ± 0	$5.2 \times 10^{-7} \pm 2.6 \times 10^{-7}$	2715 ± 237	0.15 ± 0.03
	100	Gaussian	–	30/30	0 ± 0	$5.0 \times 10^{-7} \pm 2.5 \times 10^{-7}$	4470 ± 206	0.50 ± 0.04
		Student's t	15	30/30	0 ± 0	$5.3 \times 10^{-7} \pm 2.2 \times 10^{-7}$	4190 ± 287	0.60 ± 0.04
		Student's t	1	30/30	0 ± 0	$5.3 \times 10^{-7} \pm 2.6 \times 10^{-7}$	2960 ± 226	0.34 ± 0.03
Griewank	10	Gaussian	–	30/30	0 ± 0	$4.3 \times 10^{-7} \pm 2.8 \times 10^{-7}$	2895 ± 240	0.04 ± 0.01
		Student's t	15	30/30	0 ± 0	$4.5 \times 10^{-7} \pm 2.6 \times 10^{-7}$	2870 ± 297	0.05 ± 0.02
		Student's t	1	30/30	0 ± 0	$4.7 \times 10^{-7} \pm 3.0 \times 10^{-7}$	2755 ± 472	0.04 ± 0.01
	50	Gaussian	–	30/30	0 ± 0	$6.2 \times 10^{-7} \pm 2.2 \times 10^{-7}$	3785 ± 215	0.22 ± 0.02
		Student's t	15	30/30	0 ± 0	$4.4 \times 10^{-7} \pm 2.6 \times 10^{-7}$	3605 ± 191	0.26 ± 0.03
		Student's t	1	30/30	0 ± 0	$4.5 \times 10^{-7} \pm 2.9 \times 10^{-7}$	2960 ± 284	0.18 ± 0.02
	100	Gaussian	–	30/30	0 ± 0	$6.2 \times 10^{-7} \pm 2.8 \times 10^{-7}$	4425 ± 269	0.49 ± 0.05
		Student's t	15	30/30	0 ± 0	$5.5 \times 10^{-7} \pm 2.6 \times 10^{-7}$	4250 ± 210	0.61 ± 0.03
		Student's t	1	30/30	0 ± 0	$4.5 \times 10^{-7} \pm 2.7 \times 10^{-7}$	3040 ± 270	0.36 ± 0.04
Summation Cancellation	10	Gaussian	–	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.4 \times 10^{-7}$	9165 ± 430	0.12 ± 0.01
		Student's t	15	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.6 \times 10^{-7}$	9160 ± 361	0.16 ± 0.02
		Student's t	1	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.1 \times 10^{-7}$	8960 ± 565	0.13 ± 0.01
	50	Gaussian	–	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.1 \times 10^{-7}$	$13{,}675 \pm 305$	0.78 ± 0.03
		Student's t	15	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.5 \times 10^{-7}$	$12{,}945 \pm 425$	1.00 ± 0.05
		Student's t	1	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.3 \times 10^{-7}$	$10{,}115 \pm 403$	0.63 ± 0.03
	100	Gaussian	–	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.1 \times 10^{-7}$	$18{,}185 \pm 512$	2.17 ± 0.07
		Student's t	15	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.4 \times 10^{-7}$	$16{,}555 \pm 494$	2.56 ± 0.09
		Student's t	1	30/30	0 ± 0	$-1 \times 10^{+5} \pm 2.0 \times 10^{-7}$	$10{,}770 \pm 383$	1.37 ± 0.06
Rosenbrock	10	Gaussian	–	30/30	0 ± 0	$5.0 \times 10^{-7} \pm 2.6 \times 10^{-7}$	2860 ± 394	0.04 ± 0.01
		Student's t	15	30/30	0 ± 0	$4.6 \times 10^{-7} \pm 2.8 \times 10^{-7}$	2920 ± 515	0.03 ± 0.18
		Student's t	1	26/30	78.1 ± 189	$9.6 \times 10^{-4} \pm 4.6 \times 10^{-3}$	$47{,}815 \pm 105{,}113$	0.44 ± 0.92
	50	Gaussian	–	30/30	0 ± 0	$5.3 \times 10^{-7} \pm 2.6 \times 10^{-7}$	3740 ± 216	0.20 ± 0.02
		Student's t	15	30/30	0 ± 0	$4.9 \times 10^{-7} \pm 2.6 \times 10^{-7}$	3710 ± 305	0.27 ± 0.04
		Student's t	1	30/30	0 ± 0	$5.6 \times 10^{-7} \pm 2.7 \times 10^{-7}$	3190 ± 1293	0.23 ± 1.10
	100	Gaussian	–	30/30	0.97 ± 0.4	$5.9 \times 10^{-7} \pm 2.6 \times 10^{-7}$	5220 ± 493	0.65 ± 0.06
		Student's t	15	30/30	0.87 ± 0.4	$5.6 \times 10^{-7} \pm 2.1 \times 10^{-7}$	4915 ± 509	0.68 ± 0.07
		Student's t	1	30/30	0 ± 0	$5.2 \times 10^{-7} \pm 2.9 \times 10^{-7}$	3115 ± 317	0.39 ± 0.07

*Marginal: normal; Pop size = 300; Numbers of rebels = 70.

paper has also showed that the inclusion of at least one rebel individual in the population may contribute, especially in the most difficult benchmark, to the increase in EDA-MEC performance.

Tests performed with some classic benchmark functions of numerical optimization allow a positive evaluation of the EDA-MEC. It obtained performance superior to other CEDAs in terms of success rate and number of evaluations. It even obtained great performance in the Rosenbrock function, for which only some algorithms, such as the CMA-ES, have been successful in locating its global minimum.

Although successful in these trials, including tests with up to 100 dimensions, other properties of EDA-MEC needs to be checked, such as the invariances under transformations of the search space [2]. This involves the use of shifted and

rotated benchmark functions, such as those defined in competitions on evolutionary computation [56]. It is also necessary to expand the model to other copulas, including the Archimedean, and other marginal distributions, as well as to investigate the characteristics that make a marginal appropriate for a copula in an optimization problem.

Acknowledgments

The authors gratefully acknowledge the financial support from the National Counsel of Technological and Scientific Development (CNPq) and Carlos Chagas Filho Research Support Foundation (FAPERJ).

References

[1] K. Aas, C. Czado, A. Frigessi, H. Bakken, Pair-copula constructions of multiple dependence, Insur. Math. Econ. 44 (2009) 182–198.
[2] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: IEEE Congress on Evolutionary Computation, IEEE, 2005, pp. 1769–1776.
[3] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, 1994.
[4] S. Baluja, S. Davies, Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space, in: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 30–38.
[5] E. Bonabeau, M. Dorigo, G. Théraulaz, Swarm Intelligence: from Natural to Artificial Systems, Oxford University Press, New York, 1999.
[6] J.S. De Bonet, C.L. Isbell, P. Viola, MIMIC: finding optima by estimating probability densities, Adv. Neural Inf. Process. Syst. 4 (1997) 424–430.
[7] L.B. Booker, D.E. Goldberg, J.H. Holland, Classifier systems and genetic algorithms, Artif. Intell. 40 (1989) 235–282.
[8] P.A.N. Bosman, D. Thierens, Advancing Continuous IDEAs with Mixture Distributions and Factorization Selection Metrics, in: Proceedings of the Optimization by Building and Using Probabilistic Model. OBUPM Workings in Genetic Evolution of Computational Conference GECCO–2001, Morgan Kaufmann, 2001, pp. 208–212.
[9] P.A.N. Bosman, D. Thierens, Numerical optimization with real-valued estimation-of-distribution algorithms, in: M. Pelikan, K. Sastry, E. CantúPaz (Eds.), Scalable Optimization via Probabilistic Model, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 91–120.
[10] F. Caraffini, F. Neri, L. Picinali, An analysis on separability for Memetic Computing automatic design, Inf. Sci. (Ny). 265 (2014) 1–22.
[11] U. Cherubini, E. Luciano, W. Vecchiato, Copula Methods in Finance, 1st ed., John Wiley & Sons Ltd, Chichester, 2004.
[12] L. delaOssa, J.A. Gamez, J.L. Mateo, J.M. Puerta, Avoiding premature convergence in estimation of distribution algorithms, in: 2009 IEEE Congress on Evolutionary Computing, IEEE, 2009, pp. 455–462.
[13] M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Artif. Life. 5 (1999) 137–172.
[14] P. Embrechts, A. McNeil, D. Straumann, Correlation and dependence in risk management: Properties and pitfalls, in: M. Dempster (Ed.), Risk Management Value Risk and Beyond, Cambridge University Press, 2002, pp. 176–223.
[15] M.G. Epitropakis, F. Caraffini, F. Neri, E.K. Burke, A separability prototype for automatic memes with adaptive operator selection, in: 2014 IEEE Symposium on Foundations of Computer Intelligence, IEEE, 2014, pp. 70–77.
[16] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence Through Simulated Evolution, John Wiley and Sons, New York, 1966.
[17] Y. Gao, L. Peng, F. Li, M. Liu, X. Hu, in: Z. Wen, T. Li (Eds.), Multiobjective Estimation of Distribution Algorithms Using Multivariate Archimedean Copulas and Average Ranking BT - Foundations of Intelligent Systems: Proceedings of the Eighth International Conference on Intelligent Systems and Knowledge Engineering, S, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 591–601.
[18] A.R. Goncalves, F.J. Von Zuben, Online learning in estimation of distribution algorithms for dynamic environments, in: 2011 IEEE Congress on Evolutionary Computation, IEEE, 2011, pp. 62–69.
[19] Y. González-Fernandez, M. Soto, copulaedas: An R package for estimation of distribution algorithms based on copulas (version 1), (2012).
[20] Y. González-Fernandez, M. Soto, Copulaedas: An R package for estimation of distribution algorithms based on copulas, J. Stat. Softw. 1 (9) (2014).
[21] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2001) 159–195.
[22] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, A. Auger, Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems, Appl. Soft Comput. 11 (2011) 5755–5769.
[23] G. Harik, E. Cantu-Paz, D.E. Goldberg, B.L. Miller, The gambler's ruin problem, genetic algorithms, and the sizing of populations, in: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC '97), IEEE, n.d.: pp. 7–12.
[24] G.R. Harik, F.G. Lobo, K. Sastry, Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA), Stud. Comput. Intell. 33 (2007) 39–61.
[25] M. Hauschild, M. Pelikan, An introduction and survey of estimation of distribution algorithms, Swarm Evol. Comput. 1 (2011) 111–128.
[26] M. Hyrs, J. Schwarz, Multivariate gaussian copula in estimation of distribution algorithm with model migration, in: 2014 IEEE Symposium on Foundations of Computation Intelligence, IEEE, 2014, pp. 114–119.
[27] Y. Jiao, I. Joe, Energy-efficient resource allocation for heterogeneous cognitive radio network based on two-tier crossover genetic algorithm, J. Commun. Netw. 18 (2016) 112–122.
[28] H. Joe, Multivariate Models and Dependence Concepts, Chapman and Hall, London, 1997.
[29] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, IEEE, 1995, pp. 1942–1948.
[30] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, Cambridge, 1992.
[31] P. Larrañaga, A review of estimation of distribution algorithms, in: P. Larrañaga, J.A. Lozano (Eds.), Estimation of Distribution Algorithms a New Tool Evolutionary Computation, Springer US, New York, 2002, pp. 57–100.
[32] P. Larrañaga, R. Etxeberria, J.A. Lozano, J.M. Pena, Optimization by learning and simulation of Bayesian and Gaussian networks, Tech. Rep. EHU-KZAAIK-IK-4/99, Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 1999.
[33] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Boston, 2002.
[34] J.A. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea, Towards a New Evolutionary Computation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
[35] A. Maesani, G. Iacca, D. Floreano, Memetic viability evolution for constrained optimization, IEEE Trans. Evol. Comput. 20 (2016) 125–144.
[36] S.W. Mahfoud, Niching methods for genetic algorithms, Technical Report on 95001, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana-Champaign, 1995.
[37] J.-F. Mai, M. Scherer, Simulating copulas: Stochastic models, Sampling Algorithms and Applications, Series in Quantitative Finance-Vol.4, Imperial College Press, London, 2012.
[38] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, New York, 1994.
[39] Z. Michalewicz, D.B. Fogel, How to Solve it: Modern Heuristics, 2nd ed., Springer, New York, 2000.
[40] E. Mininno, F. Neri, F. Cupertino, D. Naso, Compact differential evolution, IEEE Trans. Evol. Comput. 15 (2011) 32–54.
[41] P. Moscato, C. Cotta, A. Mendes, Memetic algorithms, in: G.C. Onwubolu, B.V Babu (Eds.), New Optimization in Technical Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 53–85.
[42] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions I. binary parameters, in: H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nat. — PPSN IV, Springer-Verlag, 1996, pp. 178–187.
[43] R.B. Nelsen, An Introduction to Copulas, 2nd ed., Springer, New York, 2006.

[44] F. Neri, E. Mininno, G. Iacca, Compact particle swarm optimization, Inf. Sci. (Ny). 239 (2013) 96–121.
[45] M. Pelikan, Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms, 1st ed., Springer-Verlag, Berlin Heidelberg, 2005.
[46] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, BOA: The bayesian optimization algorithm, in: Proceedings of the Genetic Evolutionary Computational Conference GECCO 1999, 1999, pp. 525–532.
[47] M. Pelikan, H. Mühlenbein, Marginal distributions in evolutionary algorithms, in: Proceedings of the International Conference on Genetic Algorithms Mendel'98, 1999, pp. 90–95.
[48] J.M. Peña, J.A. Lozano, P. Larrañaga, Benefits of data clustering in multimodal function optimization via EDAs, in: P. Larrañaga, J.A. Lozano (Eds.), Estimization on Distributed Algorithms a New Tool for Evolutionary Computation, Springer US, 2002, pp. 101–127.
[49] I. Rechenberg, Evolutions Strategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution, Frommann-Holzboog Verlag, Stuttgart, 1973.
[50] M.J. Reddy, P. Ganguli, Bivariate flood frequency analysis of upper godavari river flows using archimedean copulas, Water Resour. Manage. 26 (2012) 3995–4018.
[51] R. Salinas-Gutiérrez, A. Hernández-Aguirre, E.R. Villa-Diharce, Copula selection for graphical models in continuous estimation of distribution algorithms, Comput. Stat. 29 (2013) 685–713.
[52] R. Santana, Estimation of distribution algorithms with Kikuchi approximations, Evol. Comput. 13 (2005) 67–97.
[53] A. Sklar, Fonctions de répartition à n dimensions et leurs marges, Publications de l'Institut de Statistique de L'Université de Paris, 1959.
[54] M. Soto, A. Ochoa, J. Arderí, Gaussian copula estimation of distribution algorithm, Technical Report ICIMAF 2007-406, Institute of Cybernetics, Mathematics and Physics, 2007.
[55] R. Storn, K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[56] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical Report on 2005005, Nanyang Technological University, Singapore, and KanGAL, 2005.
[57] E.G. Talbi, Metaheuristics: From Design to Implementation, John Wiley and Sons, New Jersey, 2009.
[58] V. Veloso de Melo, G. Iacca, A CMA-ES-based 2-stage memetic framework for solving constrained optimization problems, in: 2014 IEEE Symposium of Foundations in Computational Intelligence, IEEE, 2014, pp. 143–150.
[59] D. Wallin, C. Ryan, On the diversity of diversity, in: 2007 IEEE Congress Evolution Computation, IEEE, 2007, pp. 95–102.
[60] S. Xiangman, T. Lixin, A novel hybrid differential evolution-estimation of distribution algorithm for dynamic optimization problem, in: 2013 IEEE Congress Evolution Computation, IEEE, 2013, pp. 1710–1717.
[61] S. Yin, X. Zhu, Intelligent particle filter and its application on fault detection of nonlinear system, IEEE Trans. Ind. Electron. 62 (2015) 1-1.
[62] A. Zhigljavsky, A. Zilinskas, Stochastic Global Optimization, Springer o, Springer, New York, 2008.