Bivariate empirical and n-variate Archimedean copulas in Estimation of Distribution Algorithms

Alfredo Cuesta-Infante, Roberto Santana, J. Ignacio Hidalgo, Concha Bielza and Pedro Larrañaga

Abstract— **This paper investigates the use of empirical and Archimedean copulas as probabilistic models of continuous estimation of distribution algorithms (EDAs). A method for learning and sampling empirical bivariate copulas to be used in the context of n-dimensional EDAs is first introduced. Then, by using Archimedean copulas instead of empirical makes possible to construct n-dimensional copulas with the same purpose. Both copula-based EDAs are compared to other known continuous EDAs on a set of 24 functions and different number of variables. Experimental results show that the proposed copula-based EDAs achieve a better behaviour than previous approaches in a 20% of the benchmark functions.**

I. Introduction

In evolutionary optimization, the class of algorithms that employ probabilistic modeling are usually called estimation of distribution algorithms (EDAs) [10], [13], [17]. In EDAs probabilistic models are learnt from the selected individuals and used to generate new solutions. This is a significant difference with respect to other evolutionary algorithms based on crossover and mutation operators. EDAs have been successfully applied to solve problems with discrete and continuous representations. Although the main rationale behind the EDAs, i.e. learning and sampling from probabilistic models, is the same for discrete and continuous problems, there exist fundamental differences between the characteristics of these two domains. In this paper we analyze a class of EDAs for problems with continuous or real value representation[1], [11].

There are many different approaches to continuous optimization using EDAs. Most of this research applies Gaussian probabilistic models [8]. However, other approximations depart from the Gaussianity assumption. Examples of these approaches include the application of independent component analysis (ICA) [3], [18], [29], histogram based probabilistic modeling [24], Cauchy distribution [19] and Copula methods [20], [27], [28]. While probabilistic modeling using Gaussian distributions has proved to be an effective alternative for many optimization problems, there are situations where Gaussian models fail. Therefore, it is a relevant question to investigate non-Gaussian approaches to probabilistic modeling in EDAs. In this paper we focus on probabilistic modeling using Copulas.

Copula functions describe the dependence structure of two or more random variables associated by a joint probability distribution function. In other words, they provide an scale-free description of how a number of random variables are distributed. Once the copula is unveiled, the whole joint probability distribution function can be found. Moreover we can use the copula function to generate new samples with such a joint probability distribution.

Bivariate, a.k.a. 2-copulas, are well known. A comprehensive text about the subject is due to Nelsen [14]. There are numerous methods to find the copula that best fits a data set [4], [5], [6] and they are being applied in many different fields such as finance [15], [16], signal processing [2] or networked systems [22]. But constructing n-copulas is difficult. Research has focused on Elliptic and Archimedean copulas because of their good properties [12].

Wang et al. [27] introduced two different bivariate copulas, and the approximation of the marginals was made by means of Gaussian distributions. Nine functions with two variables were used as benchmark and no comparison with other EDAs was included. On the other hand Salinas-Gutiérrez et al. [20], use the Archimedean family of *Frank* copulas and Gaussian copulas to model n-dimensional distributions. The introduced algorithm is inspired in the MIMIC^G_C [9] which learns a chain-shaped structure. To learn the chain, the introduced copula-based EDAs finds a permutation that minimizes the Kullback-Leibler divergence between the empirical density function and copula-based approximation. To this end, the copula entropy between each pair of variable is estimated. These copula-based EDAs were tested on five functions with $n = 10$ dimensions and results were compared with MIMIC^G_C.

Our approach is different to previous uses of copulas in EDAs in various aspects. Firstly, we use not only Archimedean or Gaussian copulas as in previous research [20], [27], [28]. Instead, we introduce empirical copulas with the purpose of not being constrained *a priori* by a set of candidate copulas. This makes necessary to develop a methodology for generating random variates no matter the dimensionality of the search. The procedure proposed in this paper does not provide neither a proper n-copula nor a joint probability distribution function of the best fitted. Instead it focuses only on obtaining a new set of points with a similar distribution, following Vapnik's motto of "trying to solve only the problem that one has, and not a more general one". Secondly, while copulas and other alternative

Alfredo Cuesta-Infante is with Felipe II College, Universidad Complutense de Madrid, Capitan 39, Aranjuez, Spain (email: acuesta@cesfelipesegundo.com).

Roberto Santana is with the Universidad Politécnica de Madrid, Campus Montegancedo, Madrid, Spain (email: roberto.santana@upm.es)

Concha Bielza and Pedro Larrañaga are with School of Computer Science of the Universidad Politécnica de Madrid, Campus Montegancedo, Madrid, Spain (email: {mcbielza,pedro.larranaga}@fi.upm.es)

J. Ignacio Hidalgo is with the Department of Computer Structure and Arquitecture (DACYA), Universidad Complutense de Madrid, Spain (email: hidalgo@dacya.ucm.es)

probabilistic modeling methods can be used together with Gaussian distributions, the approach we follow does not consider Gaussian modeling at any step. In addition we extend the use of Archimedean copulas to n variables. And finally, we conduct an extensive evaluation on a large set of benchmark functions for different dimensions.

The paper is organized as follows. Section II introduces the main concepts from copula theory. In Section III different variants of copula learning and sampling methods are proposed. The experimental framework and the numerical results from our experiments are presented in Section IV. Finally conclusions and trends for future work are outlined in Section V.

II. Copula function theory

Definition 1: A function $C(u,v) : [0,1]^2 \rightarrow [0,1]$ is a *copula* if and only if it satisfies the three following conditions:
For every $0 \le u \le 1$ and every $0 \le v \le 1$

$$C(0,v) = C(u,0) = 0 \tag{1}$$

$$C(1,v) = v, C(u,1) = u, \tag{2}$$

and for every $0 \le u_1 \le u_2 \le 1$ and every $0 \le v_1 \le v_2 \le 1$

$$C(u_2,v_2) - C(u_2,v_1) - C(u_1,v_2) + C(u_1,v_1) \ge 0. \tag{3}$$

Copulas therefore satisfy the conditions of zero-grounded bivariate distribution functions of U and V with uniform margins. Hence a probabilistic interpretation may be given in the same way as any other joint cumulative distribution function (JCDF):

$$C(u,v) = \Pr(U \le u, V \le v).$$

Then the unique joint probability density function (JPDF) $c(u,v)$ assocciated to C is such that:

$$C(u,v) = \int_{-\infty}^{u}\int_{-\infty}^{v} c(\upsilon,\nu)d\nu d\upsilon.$$

The relevance and utility of copulas is due to Sklar's theorem [23].

Theorem 1: (Sklar's Theorem) Let $H_{XY}(x,y)$ be the JCDF of X and Y with margins $F_X(x)$ and $F_Y(y)$ and such that $u = F_X(x)$ and $v = F_Y(y)$. Then there exists a copula C such that for all x, y in $(-\infty, +\infty)$,

$$H_{XY}(x,y) = C(u,v) \tag{4}$$

If F_X and F_Y are continuous then C is unique; otherwise C is uniquely determined on $Range(F_X) \times Range(F_Y)$. Conversely if C is a copula and F_X and F_Y are cumulative distribution functions (CDF) then the function H_{XY} defined by (4) is the JCDF with margins $F_X(x)$ and $F_Y(y)$.

Thus, it is possible to separate 1) the marginal behaviour due to the individual contributions of the random variables X, Y, described by its margins F_X and F_Y respectively, and 2) the dependence structure, which is given by the copula (C *couples* X and Y). Moreover, a key feature of copulas is that they are invariant under strictly monotone transformations of their random variables (U and V). In other words, the way X and Y *move together* is modelled by the copula, whatever scales of X and Y were.

Once the copula is known, one can use the conditional distribution method to generate new samples.

Definition 2: Let U and V be two random variables whose JCDF is the copula C. Then the *conditional copula* for V given $U = u$ is

$$C_u(v) = \frac{\partial}{\partial u} C(u,v). \tag{5}$$

Again, the probabilistic interpretation of the conditional copula is the same as any conditional distribution function:

$$C_u(v) = \Pr(V \le v | U = u).$$

Then, the following procedure generates 2 random variates (u,v) whose JCDF is the copula C:

1) Draw u and t, two samples uniformly distributed in the unit interval.
2) Set $v = C_u^{(-1)}(t|u_0)$.

Here $C_u^{(-1)}(t|u_0)$ denotes the quasi-inverse of the conditional copula C_u for a given $u = u_0$, defined as follows.

Definition 3: Let F be a CDF. Then the *quasi-inverse* of F is any function $F^{(-1)}$ with domain $[0,1]$ such that:

$$F(F^{(-1)}(t)) = t,$$

for all t in the $Range(F)$. Otherwise

$$F^{(-1)}(t) = \inf\{x | F(x) \ge t\} = \sup\{x | F(x) \le t\}.$$

III. Copula-based EDAs

In this section we describe two variants of copula-based EDAs (CEDA) that are proposed in this paper. The first one considers empirical copulas and the second the Archimedean family. A general pseudocode is shown in Algorithm 1. Both variants differ only in the type of learning and sampling methods used (steps *5* and *6*).

Algorithm 1: **CEDA**

1 Generate an initial population D_0 of individuals and evaluate them
2 t ← 1
3 **do** {
4 D_{t-1}^{Se} ← Select N individuals from D_{t-1} using truncation selection
5 Using D_{t-1}^{Se} as the data set, learn a copula based approximation $\mathcal{C}$
6 D_t ← Sample M individuals from $\mathcal{C}$ using a copula sampling method
7 } **until** Stopping criterion is met

A. EDAs based on empirical copulas

A straightforward approach is to use empirical distributions and numeric methods both for the derivative and the inverse functions of the copula as well as the margins. The advantage is not to be bounded to a set of candidate JCDFs nor copulas out of which one has to choose the best fitted. Thus *rare* distributions are also captured. On the other hand they have the shortcoming of rough outcomes in the extremes that numerical methods provide.

Definition 4: The *empirical CDF* of a data set $\{x_i\} \in \mathbb{R}$ with $i = 1, \ldots, N$ is the function $F(x_i) = t_i$, being

$$t_i = \frac{\#\{x_j : x_j < x_i\}}{N-1} \tag{6}$$

where the symbol $\#$ stands for the cardinality of the set.

This definition is extended to its continuous version $F(x) : \mathbb{R} \to [0, 1]$ by doing:

$$F(x) = \begin{cases} \mathtt{LIP}(x_i, t_i, x) & \text{if} \quad \inf\{x_i\} \leq x \leq \sup\{x_i\} \\ 0 & \text{if} \quad x < \inf\{x_i\} \\ 1 & \text{if} \quad x > \sup\{x_i\}, \end{cases} \tag{7}$$

where $\mathtt{LIP}(x_i, t_i, x)$ is the linear interpolation of the point x given the pairs (x_i, t_i) computed with the definition 4. The quasi-inverse of the empirical CDF is then obtained as follows:

$$F^{(-1)}(t) = \begin{cases} \mathtt{LIP}(t_i, x_i, t) & \text{if} \quad \inf\{t_i\} \leq t \leq \sup\{t_i\} \\ \inf\{x_i\} & \text{if} \quad t < \inf\{t_i\} \\ \sup\{x_i\} & \text{if} \quad t > \sup\{t_i\}. \end{cases} \tag{8}$$

Regarding the empirical copula, the starting point is to obtain its empirical density.

Definition 5: The *empirical density copula* of a data set $\{u_j, v_j\} \in [0, 1]^2$ with $j = 1, \ldots, N$, is the function $c(u_i^*, v_i^*) = t_i$, with $i = 1, \ldots, N_b$, being $N_b = 2^{\lfloor \log_2 \sqrt{N} \rfloor}$ and t_i the relative frequency of the pair (u_i^*, v_i^*) in a 3D histogram of $N_b \times N_b$ bins, each one of them centred on (u_i^*, v_i^*).

The empirical copula $C(u_i^*, v_i^*) = T_i$ is constructed doing the cumulative sum of the empirical density, first in one variable and then in the other. The extension to the continuous unit square is:

$$C(u, v) = \begin{cases} \mathtt{LIP}(u_i^*, v_i^*, T_i, u, v) \\ \quad \text{if } \inf\{u_i^*\} \leq u \leq \sup\{u_i^*\} \text{ and} \\ \quad\quad \inf\{v_i^*\} \leq v \leq \sup\{v_i^*\} \\ 0 \text{ if } u < \inf\{u_i^*\} \text{ or } v < \inf\{v_i^*\} \\ u \text{ if } v > \sup\{v_i^*\} \\ v \text{ if } u > \sup\{u_i^*\}, \end{cases} \tag{9}$$

where $\mathtt{LIP}(u_i^*, v_i^*, T_i, u, v)$ is the linear interpolation in two dimensions of the point (u, v) given the triples (u_i^*, v_i^*, T_i) computed with Definition 5.

Finally, for the conditional copula, rather than approximating the derivative with respect to the first argument of the empirical copula it is faster to take the cumulative sum of the empirical density in the direction of the second argument, already computed in the steps for obtaining the empirical copula. The outcome of this operation is a succesion of CDFs indexed by $\{u_i^*\}$ and denoted as $C_{u_i^*}(v_i^*)$. Then, alike with empirical CDFs, the extension of the inverse conditional copula to the continuous is defined by

$$C_u^{(-1)}(t) = \mathtt{EICDF}(C_{u_i^*}(v_i^*), v_i^*, t) \text{ for } u_i^* \sim u, \tag{10}$$

where the function $a_i \sim b$ returns the element of the set $\{a_i\}$ closest to b and $\mathtt{EICDF}(C_{u_i^*}(v_i^*), v_i^*, t)$ is computed with (8) using the given parameters in the inner function $\mathtt{LIP}$.

1) Learning empirical CEDA: Let $S = [a, b]^n$ be the search space. Let $P = \{p_1, \ldots, p_m\} \subset S$ be the population that is being evaluated in the objective function f. Let f_0 be the threshold that defines the subset of best fitted $X = \{x_1, \ldots, x_\ell\} \subset P$ such that $f(x_i) \geq f_0$. Representing $x_i = [x_i^1, \ldots, x_i^n]^T$, for $i = 1, \ldots, \ell$, where upperscript T denotes the transpose, the subset X takes the form of a matrix

$$X = \begin{pmatrix} x_1^1 & x_2^1 & \ldots & x_\ell^1 \\ x_1^2 & x_2^2 & \ldots & x_\ell^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & \ldots & x_\ell^n \end{pmatrix}.$$

Thus, the i-th column of X will be x_i, as defined above, whereas the j-th row of X will be denoted as $\underline{x}_j = [x_1^j, \ldots, x_\ell^j]$ for $j = 1, \ldots, n$.

Let us define the new variable y_i in the following recursive way:

$$y_i = \begin{cases} \underline{x}_1 & \text{for} \quad i = 1 \\ H_{i-1,i}(y_{i-1}, \underline{x}_i) & \text{for} \quad i = 2, \ldots, n-1, \end{cases} \tag{11}$$

where $H_{i-1,i}(y_{i-1}, \underline{x}_i)$ is the JCDF of y_{i-1} and the i-th row of X. Therefore, there must be an underlying copula $C_{i-1,i}$ that constructs its dependence structure. Thus, according to Sklar's theorem:

$$\begin{aligned} &y_1 = \underline{x}_1; \\ &y_2 = H_{1,2}(y_1, \underline{x}_2) = C_{1,2}(G_1(y_1), F_2(\underline{x}_2)) = C_{1,2}(u_1, v_2); \\ &\vdots \end{aligned}$$

In general

$$\begin{aligned} y_i \quad = \quad & H_{i-1,i}(y_{i-1}, \underline{x}_i) = \\ & C_{i-1,i}(G_{i-1}(y_{i-1}), F_i(\underline{x}_i)) = \\ & C_{i-1,i}(u_{i-1}, v_i), \end{aligned} \tag{12}$$

with $i = 1, \ldots, n-1$; being G_j the CDF of y_j and F_j the CDF of $\underline{x}_j$, for $j = 1, \ldots, n$.

2) Sampling with empirical CEDA: Now using the inverse of the conditional copula is possible to generate a whole new population $\hat{P} = \{\hat{p}_1, \ldots, \hat{p}_m\}$ with the same distribution than X following the Algorithm 2.

B. EDAs based on Archimedean copulas

As an alternative to empirical copulas one can attempt to use any of the known closed forms both for a given copula and its inverse conditional. That way neither generates a new population with a similar distribution of the old one nor obtains a *real* copula describing the dependence structure of the population. However it is possible to achieve the second objective by choosing Archimedean copulas.

Algorithm 2: **New population from Emp. CEDA**

1 Draw $\hat{u}_1$ and t, both uniformly distributed in [0,1] and obtain
$$\hat{v}_2 = \frac{\partial C_{1,2}}{\partial u}^{(-1)}(t|\hat{u}_1).$$
2 Set $\hat{p}_i^1 = G_1^{(-1)}(\hat{u}_1)$ and $\hat{p}_i^2 = F_2^{(-1)}(\hat{v}_2)$.
3 **For** $k = 2$ to $n-1$:
4 Set $\hat{y}_k = C_{k-1,k}(\hat{u}_{k-1}, \hat{v}_k)$.
5 Set $\hat{u}_k = G_k(\hat{y}_k)$.
6 Draw a new t as in step 1 and obtain
$$\hat{v}_{k+1} = \frac{\partial C_{k,k+1}}{\partial u}^{(-1)}(t|\hat{u}_k).$$
7 Set $\hat{p}_i^{k+1} = F_{k+1}^{(-1)}(\hat{v}_{k+1})$.
8 **End**
9 **Repeat** from step 1 until completing a whole new population, i.e. for $i = 1, \ldots, m$.

Algorithm 3: **New population from Arch. CEDA**

1 Draw $\hat{w}_1 = \hat{u}_1$ and t, both uniformly distributed in [0,1] and obtain
$$\hat{u}_2 = \frac{\partial}{\partial u} C^{(-1)}(t|\hat{u}_1).$$
2 Set $\hat{p}_i^1 = F_1^{(-1)}(\hat{u}_1)$ and $\hat{p}_i^2 = F_2^{(-1)}(\hat{u}_2)$.
3 **For** $k = 2$ to $n-1$:
4 Set $\hat{w}_k = C(\hat{w}_{k-1}, \hat{u}_k)$.
5 Draw a new t as in 1 and obtain
$$\hat{u}_{k+1} = \frac{\partial}{\partial u} C^{(-1)}(t|\hat{w}_k).$$
6 Set $\hat{p}_i^{k+1} = F_{k+1}^{(-1)}(\hat{u}_{k+1})$
7 **End**
8 **Repeat** from step 1 until completing a whole new population, i.e. for $i = 1, \ldots, m$.

1) Learning Archimedean CEDA:

Definition 6: An n-copula C is said to be *Archimedean* if there exists a function $\varphi(t) : [0,1] \to [0,\infty)$ continuous, strictly decreasing, convex and with $\varphi(1) = 0$ such that

$$C(u_1, u_2, \ldots) = \varphi^{(-1)}(\varphi(u_1) + \varphi(u_2) + \cdots), \qquad (13)$$

and function $\varphi(t)$ is called *generator* of C.

From Definition 6 is easy to see that Archimedean copulas are both symmetric, i.e. $C(u_1, u_2) = C(u_2, u_1)$, and associative, i.e. $C(C(u_1, u_2), u_3) = C(u_1, C(u_2, u_3)) = C(u_1, u_2, u_3)$. The latter is the key to construct the n-copula. It is important to remark the differences between previous works. Regarding [28] we extend the use to n variables; whereas, unlike [20], here it is not necessary to estimate any copula between pairs of variables. Due to the associativeness of Archimedean copulas, the copula must be exactly the same for every pair of variables. That is the reason why the learning algorithm is reduced to fixing the copula *a priori*.

2) Sampling Archimedean CEDA: What in Section III-A.2 was a recursive way only to generate new samples preserving a certain distribution now turns into the actual way to construct a real n-copula. Moreover, although the n-copula is known, it is not necessary to construct but its bivariate version. The algorithm given in Section III-A.2 is now simplified in the Algorithm 3.

Here, $C(u, v)$ is the Archimedean copula chosen in Section III-B.1 and F_i is the empirical CDF of every variable, i.e. the margins of the copula, computed as in Section III-A.

IV. Experiments

In this section we investigate the behaviour of different copula-based EDAs on a large benchmark of functions. First, the experimental framework is introduced, then, the numerical results are presented and discussed.

A. Experimental framework

We compare a number of EDAs grouped into three sets. The first and second sets are the CEDAs proposed in the paper whereas the third one is a group of well-known EDAs whose performance has been already proved. All of them follow the general scheme shown in Algorithm 1 but steps 4 and 5, where sampling and learning algorithms are implemented, are modified according to the set used.

The first set of CEDAs is based on three popular Archimedean copulas for which there are closed forms of the inverse conditional copula, namely *Frank*, *Clayton* and *HRT*, which can be found in [25] for instance. All of them depend on a parameter a that determines the degree of association of the copula and its structure of dependence as Figure 1 shows.

(A,B) Copula Frank with $a = \{-5, 5\}$. It is symmetric, allows negative dependence between the variables and tends to the independence as a goes to 0. They will be also represented as $\text{CEDA}^{Frank}_{a=\{-5,5\}}$.

(C,D) Copula HRT with $a = \{0.2, 5\}$. It is asymmetric, with greater dependence in the positive tail than in the negative, and tends to the independence as a increases. They will be also represented as $\text{CEDA}^{HRT}_{a=\{0.2,5\}}$.

(E,F) Copula Clayton with $a = \{0.2, 5\}$. It is asymmetric, with greater dependence in the negative tail than in the positive, and tends to the independence as a approaches to 0. They will be also represented as $\text{CEDA}^{Clayton}_{a=\{0.2,5\}}$.

The second set is the CEDA based on empirical copulas.

(G) Obtained by the procedures given in Section III-A and also represented as CEDA^{Emp}.

Finally, the third set consists of the following EDAs based on Gaussian distributions.

(H) UMDA_c: A univariate marginal distribution algorithm where each variable is independently modeled using a univariate Gaussian distribution [9].

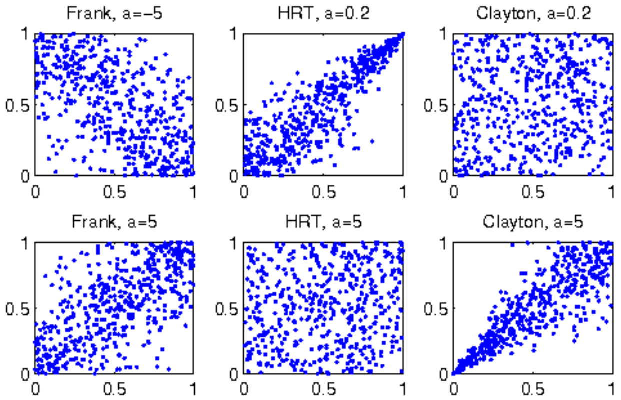

Fig. 1. Scatterplot of 500 pairs $(u, v) \in [0, 1]^2$ sampled with the used Archimedean copulas showing the way they are concentrated depending both on parameter a and the copula used.

(I) EMNA: A multivariate EDA where the density function is modeled using a multi-variate Gaussian distribution. To avoid (likely but unfrequent) numerical errors in the computation of the covariance matrix, the population is restarted when the variance goes below a given threshold [9].

(J) EDDA: The eigenspace EDA is based on an eigenspace analysis of the covariance matrix of the population. The algorithm is similar to EMNA but after an eigenspace decomposition of the covariance matrix is computed, the minimum eigenvalue is reset to the value of the maximum eigenvalue [26].

As a function benchmark we use the 24 functions from the Black-Box Optimization Benchmarking [1] at GECCO-2009. These functions as well as the experimental setup are described in [7].

The population size is $M = 1000$ for all functions and problem dimensions. The truncation selection parameter used is $T = 0.5$ and the stop criterion is reaching a maximum of 100 generations. All algorithms have been implemented in Matlab using the MATEDA-2.0 software [21]. Methods for Empirical and Archimedean copulas were developed *ad hoc*; nevertheless Matlab Statistical Toolbox 7.2 incorporates functions for all the procedures used in this paper.

B. Numerical Results

In order to evaluate the algorithms we carried out three experiments. The first one focuses on the accuracy attained as the dimension of the problem increases from 2 to 5 and to 10 variables. The second one gives a closer look to the case with 5 variables comparing also the number of successes and the number evaluations of evaluations required to achieve them. Finally, we investigate the behavior of the different EDAs from a global perspective, computing on average the fraction of successful trials as a function of the number of evaluations and of the optimization accuracy.

[1] http://coco.gforge.inria.fr/doku.php?id=bbob-2009

1) Accuracy: We consider three different dimensions: 2, 5 and 10 variables. The aim is to determine the order of magnitude of Δf, the difference between the optimum f_{opt} and the closest value of f attained, measured in ten to the power of $k \in \{-8, -4, -2, 1\}$. Results are shown in Table I where columns A to J are the values of the smallest k for each one of the EDAs used.

For each row, the smallest value is highlighted in yellow. In addition, if a CEDA (columns A to G) performs equal to the smallest non-CEDA (columns H,I,J) it is highlighted in cyan. Finally if a CEDA performs better than the best non-CEDA, it is highlighted in green.

Comparing CEDA and non-CEDA for 2 variables only for function 5 is impossible to find a CEDA that performs at least equal than a non-CEDA. The rest of functions can be split into two groups, considering the number of CEDAs that perform equal. Thus for functions $\{1, 2, 3, 4, 7, 15, 16, 17, 20, 21, 23\}$, almost half of the benchmark set, all the CEDAs attain the highest accuracy, (10^{-8}), the same than at least one non-CEDA and sometimes even better; whereas for the rest only some CEDAs perform well.

As the number of variables increases to 5 and 10 non-CEDAs become outranked in 6 functions for 5 variables and in 4 for 10 variables, and matched in other 5 functions for 5 variables and 7 for 10 variables.

2) Successful trials and number of function evaluations: We now consider not only the order of magnitude of the accuracy attained (k) but also the number of succesful trials (NT) and the median number of function evaluations to reach the best function value (RT$_{succ}$) in thousands. Additional parameters are: 5 variables, 30 trials and 198000 as the maximum number of function evaluations. Results are shown in Table II, where rows are sorted first in the decreasing order of k, then in the increasing order of NT and finally in the decreasing order of RT$_{succ}$. Thus it is clear that CEDAs outrank to non-CEDAs in a 25% of the benchmark set corresponding to functions $\{3, 4, 8, 9, 16, 23\}$, and have a quite similar performance in functions $\{1, 2\}$, which altogether cover one third of the benchmark set.

3) Empirical distribution of trials: In the last experiment we evaluate the general behavior of all the algorithms for all the functions. Figure 2 shows the empirical cumulative distribution functions (ECDFs) plotting the fraction of trials versus the number of function evaluations divided by search space dimension $D = 5$, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. The second value in the legend indicates the number of functions that were solved in at least one trial.

By analyzing the number of functions that were solved in at least one trial we get a perspective of the general behavior of the different EDAs. It can be seen that the best performance is achieved by EMNA, which is able to solve, for all levels of accuracy 10 or more functions. However, the behavior of the CEDAs is acceptable for the first level of accuracy. They are able to solve 23 out of the 24 functions. Particularly relevant is the behavior of CEDA^{Emp} which is

TABLE I

ACCURACY ATTAINED

2 Variables

f	A	B	C	D	E	F	G	H	I	J
1.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
2.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-2
3.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
4.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-3
5.	-1	-1	-1	-1	-2	-1	-1	-8	-8	-8
6.	-5	-8	-8	-5	-5	-5	-3	-8	-3	-8
7.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
8.	-5	-5	-8	-5	-5	-3	-3	-8	-8	-8
9.	-5	-5	-5	-5	-5	-8	-5	-8	-8	-8
10.	-3	-1	-1	-3	-1	-1	-1	-5	-8	-3
11.	-3	-3	-2	-1	-2	-2	0	-3	-8	-2
12.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-3
13.	-3	-3	-3	-3	-5	-3	-2	-3	-8	-3
14.	-8	-5	-5	-8	-8	-8	-3	-8	-8	-5
15.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
16.	-8	-8	-8	-8	-8	-8	-8	-5	-8	-3
17.	-8	-8	-5	-8	-8	-5	-8	-8	-5	-8
18.	-3	-2	-1	-2	-3	-3	-2	-8	-8	-3
19.	-5	-8	-5	-5	-8	-5	-5	-8	-8	-8
20.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
21.	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
22.	-8	-8	-8	-8	-8	-8	-5	-8	-8	-8
23.	-2	-1	-1	-1	-2	-1	-1	-1	-2	-1
24.	0	0	-1	0	0	0	-1	-1	-1	0

5 Variables

f	A	B	C	D	E	F	G	H	I	J
1.	-8	-8	-8	-5	-5	-5	-8	-8	-8	-8
2.	-8	-8	-8	-3	-2	-3	-8	-8	-8	1
3.	-8	-8	-5	-3	-5	-5	-8	0	1	1
4.	-8	-8	-2	-3	-2	-2	-5	1	1	1
5.	-1	0	0	0	0	0	0	-8	-8	-8
6.	0	0	0	0	0	-1	0	0	0	-8
7.	-1	-5	-2	0	-1	0	-1	-8	-8	-8
8.	0	0	0	1	0	1	-1	1	0	1
9.	0	0	0	1	0	1	-1	1	0	0
10.	0							0	-8	1
11.	1	1	1	1	1	1	1	0	-8	1
12.	-1	0	-1	0	-1	0	-1	0	-5	1
13.	-2	-1	0	0	0	0	0	-3	-8	-1
14.	-3	-3	-3	-2	-3	-2	-3	-5	-8	-3
15.	1	1	1	1	1	1	0	1	0	1
16.	-1	-1	0	0	0	0	0	0	0	0
17.	-3	-2	-3	-3	-2	-2	-2	-8	-5	-8
18.	0	-1	-1	0	0	0	-1	-3	-5	-3
19.	0	0	0	0	-1	0	-1	-1	-1	-1
20.	-5	0	0	0	-3	0	-1	-8	0	0
21.	0	0	-1	0	-2	-1	-5	-8	-8	-8
22.	-3	-3	-5	0	-3	-1	-1	-5	-8	-8
23.	-1	0	0	0	0	0	0	0	0	0
24.	1	1	1	1	1	1	1	1	1	1

10 Variables

f	A	B	C	D	E	F	G	H	I	J
1.	-5	-5	-1	-2	-1	-3	-8	-8	-8	-8
2.	-3	-2		0		0	-3	-8	-8	
3.	-2	-3	0	0	1	0	-5			
4.	-2	-1	1	1	1	1	-1			
5.	0	0	0	1	0	1	0	-8		-8
6.	0	0					1	1		-5
7.	0	0	0	0	0	1	-8	-1	-8	-8
8.	0	0	0	1	1	1	0	1	1	1
9.	1	1		1		1	1	1	1	1
10.									-8	
11.							1		-8	
12.	1						-1	0	-2	1
13.						1	0	0	-8	0
14.	-2	-2	0	-1	0	-1	-3	-3	-8	-3
15.							1		1	0
16.	1	1	1	1	1	1	1	1	1	1
17.	1	1	1	0	1	0	-1	-3	-5	-5
18.	1	1	1	0	1	0	-1	-2	-5	-2
19.	1	1	1	1	1	1	1	0	0	0
20.	0	0	1	0	1	1	0	1	1	1
21.	1	1	1	1	1	-1	-3	-5	-8	-8
22.	0	0	1	1	1	1	0	0	0	0
23.	0	0	0	0	1	0	0	0	0	0
24.										

Accuracy attained by the EDAs proposed, represented in 10^k where k is the value shown for each function (rows $1-24$) and each EDA (columns A-J). Blank spaces mean that the algorithm did not achieve an accuracy with order of magnitude 10^1. Tables are shown considering 2, 5 and 10 variables in the objective functions f. The smallest value of each row is highlighted in yellow. Values in columns A to G equal to the smallest of columns H, I, and J are highlighted in cyan. Values in columns A to G smaller than the smallest of columns H, I, and J are highlighted in green.

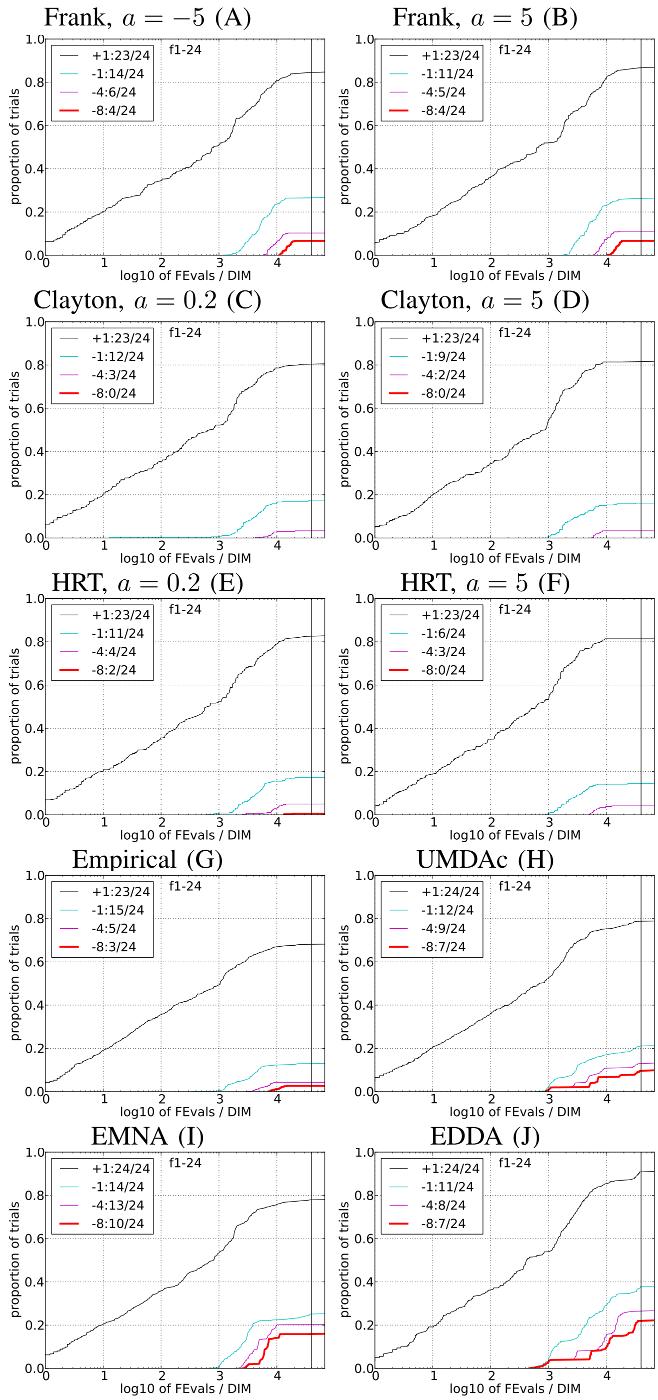

Fig. 2. Empirical cumulative distribution functions (ECDFs) plotting the fraction of trials versus the number of function evaluations divided by search space dimension $D = 5$, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. The second values in the legends indicates the number of functions that were solved in at least one trial.

the only algorithm that solves 15 out of the 24 for the second level of accuracy. There are important differences between the Archimedean CEDAs, particularly for the last level of accuracy where CEDA$^{Frank}_{a=-5}$ and CEDA$^{Frank}_{a=5}$ are the only Archimedean CEDAs that solve at least one function.

Considering the fraction of successful trials (those were the desired accuracy was reached), the CEDAs using Frank copulas reach better results than EMNA and UMDAc, but in this case the best contender is EDDA.

TABLE II

PERFORMANCE FOR 5 VARIABLES AND EVERY OBJECTIVE FUNCTION

$f.1$				$f.2$				$f.3$				$f.4$				$f.5$				$f.6$			
EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc
H	-8	15	26	H	-8	15	36	B	-8	8	250	A	-8	1	2800	J	-8	15	4.3	J	-8	5	530
J	-8	15	27	I	-8	15	230	A	-8	5	470	B	-8	1	2800	H	-8	15	30	F	-1	1	2800
I	-8	14	250	A	-8	9	220	G	-8	3	58	G	-5	3	34	I	-8	1	5900	D	0	4	550
A	-8	9	190	G	-8	8	61	F	-5	2	1300	D	-3	1	2800	A	-1	1	2800	G	0	3	31
G	-8	8	41	B	-8	7	310	E	-5	1	2800	E	-1	4	580	G	0	18	15	A	0	3	820
B	-8	8	240	C	-8	1	2900	C	-5	1	2800	C	-1	3	2800	B	0	15	7.8	B	0	3	820
C	-8	1	2800	F	-3	3	840	D	-3	2	1300	F	-1	2	1300	E	0	14	25	I	0	3	1900
F	-5	4	580	D	-3	1	2800	H	0	3	85	H	1	16	20	D	0	14	32	C	0	2	820
E	-5	4	590	E	-1	1	2800	I	1	17	170	J	1	15	26	F	0	12	65	E	0	2	1300
D	-5	4	590	J	1	3	960	J	1	15	12	I	1	15	220	C	0	2	25	H	0	1	31

$f.7$				$f.8$				$f.9$				$f.10$				$f.11$				$f.12$			
EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc
J	-8	15	52	G	-1	1	19	G	-1	1	19	I	-8	15	230	I	-8	15	230	I	-5	1	5800
I	-8	15	210	A	0	3	820	A	0	2	1300	H	0	1	140	H	0	3	180	C	-1	3	2800
H	-8	1	33	E	0	2	1300	B	0	2	1300	J	1	4	690	G	1	14	33	G	-1	1	72
B	-5	1	2800	B	0	2	1300	C	0	2	2900	G				J	1	14	62	E	-1	1	2800
G	-1	4	20	C	0	2	2800	J	0	1	2800	E				D	1	8	180	A	-1	1	2800
A	-1	4	580	I	0	2	2900	E	0	1	2800	F				F	1	7	230	B	0	4	600
C	-1	3	1300	D	1	17	22	I	0	1	5900	C				B	1	5	410	H	0	3	30
E	-1	1	2800	H	1	16	9.6	H	1	19	36	D				E	1	4	550	F	0	3	830
F	0	11	83	J	1	15	7.2	D	1	14	27	A				C	1	1	310	D	0	3	830
D	0	7	240	F	1	14	22	F	1	13	45	B				A	1	1	2800	J	1	7	370

$f.13$				$f.14$				$f.15$				$f.16$				$f.17$				$f.18$			
EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc
I	-8	12	350	I	-8	15	230	I	0	4	1400	A	-1	1	2800	J	-8	15	150	I	-5	12	350
J	-1	9	300	H	-5	13	90	G	0	1	37	B	-1	1	2800	H	-8	7	85	J	-3	13	220
H	-1	2	37	B	-5	5	440	H	1	16	11	G	0	16	58	I	-5	15	250	H	-3	1	11
A	-1	2	1300	J	-5	3	910	J	1	15	12	I	0	13	360	C	-3	4	2900	C	-1	3	1400
B	-1	1	2900	A	-3	5	450	A	1	15	54	H	0	12	72	G	-1	13	28	B	-1	2	1300
E	0	4	590	C	-3	4	2800	B	1	14	74	E	0	5	430	B	-1	9	190	G	-1	1	110
D	0	3	830	G	-3	3	61	E	1	8	210	J	0	5	480	A	-1	8	220	D	0	12	64
G	0	2	62	E	-3	1	2800	F	1	4	560	D	0	3	810	D	-1	6	330	F	0	12	66
F	0	2	1300	F	-1	15	8.6	D	1	2	1300	C	0	2	830	F	-1	5	410	A	0	12	88
C	0	2	1300	D	-1	15	9.6	C	1	1	280	F	0	2	1300	E	-1	4	570	E	0	8	200

$f.19$				$f.20$				$f.21$				$f.22$				$f.23$				$f.24$			
EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc	EDA	k	NT	RTsucc
I	-1	5	1100	H	-8	15	5.5	J	-8	12	120	J	-8	3	960	A	-1	1	2800	H	1	17	39
H	-1	3	57	A	-5	1	2800	I	-8	10	430	I	-8	3	1800	G	0	30	3.3	J	1	15	28
J	-1	3	880	E	-3	1	2800	H	-8	3	49	C	-5	5	2800	I	0	30	26	G	1	15	31
G	-1	2	110	G	-1	1	21	G	-5	1	43	H	-5	3	5.8	H	0	30	96	I	1	15	230
E	-1	1	3000	I	0	16	220	C	-1	3	2800	E	-3	6	3400	E	0	15	9.6	B	1	8	260
D	0	15	5.2	B	0	15	16	E	-1	1	2800	B	-3	1	2800	B	0	15	17	A	1	6	370
F	0	15	7.7	J	0	14	110	F	-1	1	2800	A	-3	1	2900	J	0	15	44	E	1	5	430
A	0	15	14	F	0	13	46	A	0	8	190	G	-1	4	5.9	D	0	14	25	F	1	5	430
B	0	15	18	D	0	10	110	B	0	8	190	F	-1	3	870	F	0	13	43	D	1	3	820
C	0	2	12	C	0	2	140	D	0	6	300	D	0	12	55	C	0	2	72	C	1	1	230

Performance of the proposed EDAs for each objective function (in all cases $n = 5$). Tables are sorted first by increasing Δf, measured in 10^k and represented by k. The second sorting criterion is the number of successfull trials (NT) in decreasing order. Finally we also consider the number of function evaluations necessary to reach the optimum (RTsucc) in increasing order.

V. CONCLUSIONS

In this paper we first introduce a simple method to use bivariate empirical copulas in n-dimensional EDAs. Afterwards we change the learning method by using an Archimedean copula instead of the empirical one. Three Archimedean copulas, with different parameters, have been used to model search distributions in EDAs. We have compared CEDA algorithms with classical EDAs based on Gaussian distributions in a wide set of functions, representing different domains of difficulty.

The use of copula functions in EDAs is still a large unexplored area. Previous works in this line either consider only 2-dimensional problems or attempt to find an underlying n-copula that captures the real dependence structure. Our proposal is to consider bivariate empirical copulas first. Using them we do not provide the real underlying n-copula, but a method to generate new populations considering the marginal behaviour of each variable and the dependence between variable x_j with the distribution of the copula $C_{j-1,j-2}$. Then, due to Archimedean copulas are associative, using one of them instead of the empirical copulas turns the method proposed into a way to construct real n-copulas. Again it is not the real underlying copula because it needs to be prefixed beforehand. In that sense, our proposal is similar to standard EDAs that rely on Gaussian multivariate distribution to obtain new generations, which is seldom the one that models the real distribution. The advantage of our proposal consists of not only allowing the incorporation of marginal behaviours but also the provision of closed forms of the inverse conditional copula, so faster and more accurate outcomes are expected.

Our hypothesis is that there are situations where the assumption of Gaussian distributions can be outranked by best fitted distributions. The exhaustive experimental results show that CEDAs proposed outperformed Gaussian EDAs for a noticeable 20% of the tested functions, whereas in the remaining 80% it is still possible to find better performance in some of the CEDAs proposed with respect to at least one non-CEDA. Another result derived from our analysis is the evidence that the parameters of the Archimedean copulas can have an important effect in the optimization results.

Next steps in our research are: 1) To determine which characteristics of CEDAs make them particularly suitable to optimize a given function. 2) Neither a study of which bivariate Archimedean copula nor which parameter is best for conducting the CEDA has been done. Thus, it is interesting to investigate whether tuning copula parameters may improve CEDA performance and the computational burden that it involves. 3) To incorporate advanced sampling strategies in order to avoid search stagnation.

This research is part of the CajalBlueBrain project. It has been partially supported by TIN-2008-06815-C02-02, TIN2007-62626 and TIN 2008-00508; Consolider Ingenio 2010 - CSD2007-00018 and 2010 2007/2011 projects (Spanish Ministry of Science and Innovation); by Interligare Institute for Innovation in Intelligence (I4) and also by the Real Colegio Complutense at Harvard.

[1] R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J. L. Flores, J. A. Lozano, Y. Van de Peer, R. Blanco, V. Robles, C. Bielza, and P. Larrañaga. A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, 1(6):doi:10.1186/1756–0381–1–6, 2008.

[2] N. Brunel and W. Pieczynski. Unsupervised signal using hidden Markov chains with copulas. *Signal processing*, 85:2304–2315, 2005.

[3] D. Cho and B. Zhang. Evolutionary continuous optimization by distribution estimation with variational Bayesian independent component analyzers mixture model. In *Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242, pages 212–221. Springer, 2004.

[4] E. W. Frees and E. A. Valdez. Understanding relationships using copulas. *North American Actuarial Journal*, 2:1–25, 1998.

[5] C. Genest, E. Masiello, and K. Tribouley. Estimating copula densities through wavelets. *Insurance: Mathematics and Economics*, 44(2):170 – 181, 2009.

[6] C. Genest and L. P. Rivest. Statistical inference procedures for bivariate Archimedean copulas. *Journal of the American Statistical Association*, 88(423):1034–1043, September 1993.

[7] N. Hansen, A. Auger, S. Finck, and R. Raymond. Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup. Technical Report RR-6828, INRIA, 2009.

[8] S. Kern, S. D. Müller, N. Hansen, D. Büche, J. Ocenasek, and P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms– a comparative review. *Natural Computing*, 3(1):77–112, 2004.

[9] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report EHU-KZAA-IK-4/99, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 1999.

[10] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[11] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Springer, 2006.

[12] A. J. McNeil. Sampling nested Archimedean copulas. *Journal of Statistical Computation and Simulation*, 78(6):567–581, 2008.

[13] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lectures Notes in Computer Science*, pages 178–187, Berlin, 1996. Springer.

[14] R. B. Nelsen. *An Introduction to Copulas*. Lecture Notes in Statistics. Springer, 2nd edition, 2006.

[15] F. L. P. Embrechts and A. McNeil. *Modelling Dependence with Copulas and Applications to Risk Management (Handbook of Heavy Tailed Distributions in Finance)*. S. Rachev, 2003.

[16] A. J. Patton. Copula-based models for financial time series. OFRC working papers series, Oxford Financial Research Centre, 2008.

[17] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[18] P. Pošík. *On the use of probabilistic models and coordinate transforms in real-valued evolutionary algorithms*. PhD thesis, Faculty of Electrical Engineering. Czech Technical University in Prague, Prague, Czech Republic, 2007.

[19] P. Pošík. BBOB-benchmarking a simple estimation of distribution algorithm with Cauchy distribution. In *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference GECCO-2009*, pages 2309–2314, New York, NY, USA, 2009. ACM.

[20] R. Salinas-Gutiérrez, A. Hernández-Aguirre, and E. R. Villa-Diharce. Using copulas in estimation of distribution algorithms. In *MICAI 2009: Advances in Artificial Intelligence*, volume 5845 of *Lecture Notes in Computer Science*, pages 658–668, Guanajato, Mexico, 2009. Springer.

[21] R. Santana, C. Bielza, P. Larrañaga, J. A. Lozano, C. Echegoyen, A. Mendiburu, R. Armañanzas, and S. Shakya. MATEDA: A Matlab package for the implementation and analysis of estimation of distribution algorithms. *Journal of Statistical Software*, 2010.

[22] N. D. Singpurwalla and C. Kong. Specifying interdependence in networked systems. *IEEE Trans. on Reliability*, 53(3):401–405, 2004.

[23] A. Sklar. Fonctions de repartition a n dimensions et leurs marges. *Publ. Inst. Statist (Univ. Paris)*, 8, 1959.

[24] S. Tsutsui, M. Pelikan, and D. E. Goldberg. Evolutionary algorithm using marginal histogram in continuous domain. In *Optimization by Building and Using Probabilistic Models (OBUPM) 2001*, pages 230–233, San Francisco, California, USA, 7 2001.

[25] G. Venter. Tails of copulas. In *Proc. of ASTIN Collquium of International Actuarial Association, Washington*, 2001.

[26] M. Wagner, A. Auger, and M. Schoenauer. EEDA: A new robust estimation of distribution algorithm. Technical Report RR-5190, INRIA, 2004.

[27] L.-F. Wang, J.-C. Zeng, and Y. Hong. Estimation of distribution algorithm based on Archimedean copulas. In *Proceedings of the International Conference GEC'2009*, pages 993–996, Shangai, China, 2009. ACM.

[28] L.-F. Wang, J.-C. Zeng, and Y. Hong. Estimation of distribution algorithm based on copula theory. In *Proceedings of the 2009 Congress on Evolutionary Computation CEC-2009*, pages 1057–1063, Norway, 2009. IEEE Press.

[29] Q. Zhang, N. M. Allinson, and H. Yin. Population optimization algorithm based on ICA. In *Proceedings of the 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 33–36, 2000.