A Posteriori Pareto Front Diversification Using a Copula-Based Estimation of Distribution Algorithm

Abdelhakim Cheriet
LESIA Laboratory, Biskra University
Algeria

Foudil Cherif
LESIA Laboratory, Biskra University
Algeria

***Abstract*—We propose CEDA, a Copula-based Estimation of Distribution Algorithm, to increase the size, achieve high diversity and convergence of optimal solutions for a multiobjective optimization problem. The algorithm exploits the statistical properties of Copulas to produce new solutions from the existing ones through the estimation of their distribution. CEDA starts by taking initial solutions provided by any MOEA (Multi Objective Evolutionary Algorithm), construct Copulas to estimate their distribution, and uses the constructed Copulas to generate new solutions. This design saves CEDA the need of running an MOEA every time alternative solutions are requested by a Decision Maker when the found solutions are not satisfactory. CEDA was tested on a set of benchmark problems traditionally used by the community, namely UF1, UF2, ..., UF10 and CF1, CF2, ..., CF10. CEDA used along with SPEA2 and NSGA2 as two examples of MOEA thus resulting in two variants CEDA-SPEA2 and CEDA-NSGA2 and compare them with SPEA2 and NSGA2. The results of The experiments show that, with both variants of CEDA, new solutions can be generated in a significantly smaller without compromising quality compared to those found SPEA2 and NSGA2.**

Keywords*—Multiobjective Optimization Problems; Evolutionary Algorithms; Estimation of Distribution Algorithms; Copulas

I. INTRODUCTION

A Multiobjective Optimization Problem (MOP) is an optimization problem that involves multiple functions with objectives that need to be optimized simultaneously. These objectives are usually contradictory so much so improving one objective may degrade many others. Under these circumstances, there does not exist a single solution that optimizes all functions. Instead, there typically are a number of optimal solutions, called Pareto solutions, which are considered equally good and cannot be ordered completely [1].

Although these Pareto solutions are considered equally good, a decision maker involved in working with the Pareto solutions obtained from solving a multiobjective problem may not be satisfied with some of them. In many of these cases, a decision maker may need to solve the multiobjective problem again with the expectation of finding another set of solutions that suit his needs in a better way.

Searching for new solutions by running a multiobjective problem solver each time may not be practical as finding a new solution can be complex and require a significant amount of time and resources, particularly if the solution technique used is not appropriate.

Motivating by the effort to make it more efficient for a decision maker to search for new solutions, this paper target reducing the time needed to generate new solutions without compromising their qualities. This paper propose, a Copula-based Estimation of Distribution Algorithm. CEDA belongs to the class of Estimation of Distribution Algorithms (EDA) [2], which is itself a class of Evolutionary Algorithms (EA) [3] usually used to solve multiobjective problems. In contrast to EA where new solutions are generated using an implicit distribution defined by one or more variation operator (mutation, crossover), EDA uses an explicit probability distribution model to characterize the interactions between the solutions. This feature along with their good global searching ability makes EDA well suited for efficiently generating new solutions.

Although there are many variants of EDA (See Section 4 for details), the work (CEDA) based on Copulas [4] for their ability to provide a scale-free description of how Pareto solutions are distributed. With Copulas, a joint probability distribution function can be constructed which makes is particularly easy to generate new sample solutions according to that joint probability distribution function. This makes CEDA efficient in generating new solutions in quick way with a high degree of quality thereby making it convenient for a decision maker to search for new solutions that would better suit his needs.

Briefly, this is achieved by the way CEDA operates, which starts by selecting the best individual using a MOEA (Multi Objective Evolutionary Algorithm)[5,6,7,8] from a population generated randomly. Then, CEDA uses the selected individuals to estimate their distribution using a Copula. The constructed Copula is used to generate a new population. CEDA continues with generating and selecting the best individuals until the stop condition is met. When CEDA stops, the latest generated individuals are considered Pareto optimal solutions and the last Copulas can be used in later calls of CEDA to generate alternative optimal solutions if those generated do not satisfy the needs of the Decision Maker. This design saves CEDA the need of running an MOEA every time alternative solutions are requested by a Decision Maker when the found solutions are not satisfactory.

The main contributions of this paper are the following:

- Devise a Copula-based EDA to increase the size, deliver high diversity, and achieve a quick convergence of Pareto optimal solutions for a multiobjective optimization problem. We achieve this by exploiting the

statistical properties of Copulas to produce new solutions from the existing ones through the estimation of their distribution.

- Define a new performance metric called solution generation efficiency to measure the speed of generating new Pareto optimal solutions in terms of the number of objective function evaluations.
- Thoroughly test CEDA on a set of benchmark problems traditionally used by the community, namely UF1,…,UF10, CF1,…,CF10, using SPEA2 [6] and NSGA2 [5] algorithms as two example candidates for MOEA selecting methods. Finding that new Pareto optimal solutions can be generated in a significantly smaller time compared to those found by NSGA2 and SPEA2, without compromising the quality (convergence and diversity) of these solutions.

The rest of the paper is organized as follows. In Section 2 a definition of multiobjective optimization problems is given. In Section 3, provide an overview of the work carried out in the area of multiobjective optimization. In Section 4, an overview on EDA is presented and described how they generally operate. In Section 5, provide a mathematical definition of Copula and some of their features used in EDA. We present the contribution CEDA Copula-based EDA in Section 6 and evaluate its performance on various benchmark problems in Section 7. We conclude our paper in Section 8.

II. Multiobjective Optimization

A multi-objective optimization problem is an optimization problem that involves multiple objective functions [1]. In mathematical terms, a multi-objective optimization problem can be formulated as follows.

$$\begin{aligned} &\min F(x) && \text{where } F = (f_0, f_1, \ldots, f_m) \\ &\text{subject to } G(x) \leq 0 \end{aligned} \quad (1)$$

With $\mathbf{x} \in \mathbb{R}^n, \mathbf{F}(\mathbf{x}) \in \mathbb{R}^m, \mathbf{G}(\mathbf{x}) \in \mathbb{R}^p$, we have m functions to optimize and p constraints to satisfy. The main goal of optimization methods is to find an optimal solution for the problem described in (1). Note that a multiobjective problem has many objectives to achieve which are usually mutually contradictory. Therefore, a relation for assessing the goodness of a solution compared to another one should be defined. Typically, the Pareto Dominance relation (see Definitions below) is used to achieve this end.

Definition 1 Considering a minimization problem, a decision vector **u weakly dominates v** ($\mathbf{u} \preccurlyeq \mathbf{v}$) *iff*

$$f_i(u) \leq f_i(v), \forall i \in \{0,1,\ldots,m\} \text{ and}$$
$$\exists j \in \{0,1,\ldots,m\}, f_j(u) < f_j(v).$$

Definition 2 Considering a minimization problem, a decision vector **u dominates v** ($\mathbf{u} \prec \mathbf{v}$) *iff*

$$f_i(u) < f_i(v), \forall i \in \{0,1,\ldots,m\}.$$

Definition 3 A solution x^* is a **Pareto optimal** solution if and only if there is no other admissible solution x where f(x) dominates $f(x^*)$. So the solution of a Multiobjective problem is a set of solutions which are not dominated by any other solution, we call this set the **Pareto Solutions *PS***. The image of this set in the objective space form the **Pareto Front *PF***:

III. Related Work

There are in the literature many methods for solving a multiobjective problem. Those based on EA, referred to as MOEA (Multiobjective Optimization Evolutionary Algorithms) are among the most used ones due to the good quality/cost trade-off of the solutions they provide [9]. MOEA may be classified according to the following aspects: (1) the techniques used to solve the optimization problem and (2) the schemes used for the reproduction of the offspring.

In the MOEA based on decomposition MOEA/D [9], the multiobjective problem (MOP) is decomposed into a number of scalar objective optimizations (SOPs). The objective of each SOP is called sub-problem. The population is composed in every generation with the best solution found for each sub-problem [10].

The Indicator-based MOEA framework is a recent kind of resolution which uses the Quality Indicator of the approximated Pareto Front to guide the search, the Generational Distance and the Hypervolume are two examples of the indicators used in the work of [11, 12, 13].

Another type of the MOEA frameworks is the one that is based on preference. In this class of framework, the Decision Maker (DM) is involved in the choice of preferred solutions, so the MOEA method needs to get a Pareto Front of interest to the DM. Various algorithms exist according to the way of involving the DM, a priori, a posteriori, or interactively.

In many a priori approaches (e.g. [14]), a preference point or region is given to guide the search for solutions process. The preference points are chosen according to the DM demands. After getting the preferred direction, the search process is executed from the begin to the end without involving the DM. Note that the solution obtained after executing the algorithm is usually not the best solution and may not even be close to the most preferred solution.

In a posteriori methods (e.g. NSGAII [5], SPEA2 [6]), optimal solutions are obtained using an evolutionary algorithm ignoring the interaction with the DM. After getting the PS, the DM can choose one of the obtained solutions. A posteriori methods do not provide the DM with the option of guiding the search for new solutions thereby possibly leading to solutions that are not of interest to the DM.

In interactive methods (e.g. [15, 16]), the DM directs the search for new solutions with the aim for finding solution that are of interest to them. Although these methods help the DM find good solutions to their problem, the interaction process significantly slows down the computation of solutions.

MOEA can also be classified according to the method they use for reproduction (e.g. the DE (Differential Evolution)-based algorithms [17], the Immune-based algorithms [18] , the

PSO (Particle swarm optimization)-based [19] algorithms, and the probabilistic model-based algorithms).

The probabilistic model-based approaches are considered as a new paradigm in the evolutionary computation. Their principal idea is extracting the statistical information from their previous generations and trying to build a probabilistic distribution model of the best candidate solutions. This distribution is used to sample new individuals (solutions). Examples of probabilistic model-based algorithms include those using Ant Colony Optimization, Cross Entropy [20], and Quantum-inspired Genetic Algorithm [21].

Another very important class of the probabilistic-based models are those based on the estimation of distribution, known as the EDA (Estimation of Distribution Algorithms). This class of algorithms was first introduced by Mühlenbein and Paaß [22]. The rest of this paper mainly deals with this class of algorithms, which will be explained in details in Section 4.

IV. Estimation of Distribution Algorithms

The Estimation of Distribution Algorithms is a class of Evolutionary Algorithms. It is a population based algorithm which starts with an initial population usually a random one, and then tries to select the best solutions using a fitness function (for example in the experimentation, the best solution is the one that is not dominated by any another solution). The statistical properties of the selected solutions (individuals) are used to find a distribution or a kind of function or law representing all the selected solutions. The EDA algorithms try in every generation of the algorithm to estimate the distribution of the best solution in this generation. After finding or estimating the distribution of the best-selected solutions, a number of new individuals are generated using the created function or law. In general, those new individuals have the same proprieties of the best solutions of the precedent generation. The algorithm runs many generations according to the steps described above until a criterion stop is achieved [2].

The general steps followed by an EDA are described in Algorithm 1:

Algorithm 1 Estimation of Distribution Algorithm

Initialization

While Not termination criteria **do**

Select best Solutions

Estimate the best Solutions Distribution

Generate a candidate Solutions

End While

Most of Estimation of Distribution Algorithms may be classified into two categories: those that deal with discrete variables and those that deal with the real-valued vectors. In the discrete variables class, we find algorithms that use univariate models, which assume that the problem variables are independent. Under this assumption, the probability distribution of any individual variable should not depend on the values of any other variables.

Mathematically, a univariate model decomposes the probability of a candidate solution $(X_1, X_2, \cdots, X_n)$ into the product of probabilities of individual variables as

$$p(X_1, X_2, \cdots, X_n) = p(X_1)p(X_2)\cdots p(X_n)$$

where $p(X_i)$ is the probability of variable X_i, and $p(X_1, X_2, \cdots, X_n)$ is the probability of the candidate solution $(X_1, X_2, \cdots, X_n)$. One of simplest algorithms that uses this idea is the Univariate Marginal Distribution Algorithm (UMD). UMDA works on binary strings and uses the probability vector $p = (p_1, p_2, \cdots, p_n)$ as the probabilistic model, where p_i denotes the probability of a "1" at position i of solution strings.

One of the main drawbacks of UMDA is the necessity of keeping the selected individuals to calculate the probability vector. To alleviate this problem, Incremental EDAs propose to update the probability vector incrementally to avoid keeping the list of all individuals. Population-Based Incremental Learning (PBIL) is an example of Incremental EDAs where probability vector elements are calculated according to the following equation:

$$p_i = (p_i * (1.0 - LR)) + (LR * v_i)$$

where p_i is the probability of generating a 1 in bit at position i, v_i is the ith position in the solution string and LR is the Learning Rate specified by the user. Although using univariate models is efficient particularly in saving memory usage, the assumption that problem variables are independent will often prevent efficient convergence to the optimum when problem variables interact strongly.

Tree-based models are another EDAs that deal with discrete variables. This type of EDAs is capable of capturing some pair-wise interactions between variables. In tree-based models, the conditional probability of a variable may only depend on at most one other variable. The Mutual-Information-Maximizing Input Clustering (MIMIC) uses a chain distribution to model interactions between variables. Given a permutation of the n variables in a problem, $\pi = i_1, i_2, \cdots, i_n$, MIMIC decomposes the probability distribution of $p(X_1, X_2, \cdots, X_n)$ as

$$p_\pi(X) = p(X_{i_1}|X_{i_2})p(X_{i_2}|X_{i_3})\cdots p(X_{i_{n-1}}|X_{i_n})p(X_{i_n})$$

where $p(X_{i_j}|X_{i_{j+1}})$ denotes the conditional probability of X_{i_j} given $X_{i_{j+1}}$.

All EDAs motioned previously are applicable to problems with candidate solutions represented by fixed-length strings over a finite alphabet. However, candidate solutions for many problems are represented using real-valued vectors. In these problems, the variables cover an infinite domain so it is no longer possible to enumerate variables' values and their probabilities. This gives rise to EDAs that deal with the real-coded values. One example of dealing with the real-coded values is to manipulate these through discretization and variation operators based on a discrete representation. Typically, there are three different methods of discretization: fixed-height histograms, fixed-width histograms, and k-means clustering.

The next algorithms are examples of EDAs that work directly with the real-valued variables themselves. The Estimation of Gaussian Networks Algorithm (EGNA) works by creating a Gaussian network to model the interactions between variables in the selected population of solutions in each generation [2].

Recently a new approach to developing EDAs to solve real-valued optimization problem has been developed that is based on Copula theory. The main idea of Copulas is to decompose the multivariate joint distribution into each univariate distribution and a Copula. Copula is a function that embodies the relationship of the variables [23, 24]. The use of Copula-based models in continuous EDAs places these algorithms in an advantageous position in comparison with other EDAs that rely on the assumption of a particular multivariate distribution, such as the multivariate normal distribution [25, 26]. By means of Copulas, any multivariate distribution can be decomposed into the marginal distribution and the Copula that determines the dependence structure between the variables.

The main steps of a Copula-based EDA are resumed in the Algorithm 1.a:

Algorithm 1.a Copula-based EDA

Generate initial population P_0

t = 1

While not stop criterion do

P_t^s=select best individual

Use P_t^s to learn (estimate parameters of) a Copula C

P_{t+1}^s= sample individuals from C

End while

Many types of Copula have been used in the literature. In [26], the authors used T-Copula, in [27, 28, 29, 30] , the authors used an Archimedean Copula, in [31] the authors used Clayton Copula, and in [32, 33, 34] , the authors combined more than one Copula to find the best estimation. This paper, will focus on Archimedean Copulas for their ability to model dependence in high dimensions with only one parameter, which has the good effect of speeding up multiobjective optimization computation time.

V. Mathematical Overview on Archimedean Copulas

As defined in [4], Copulas are functions that join or couple multivariate distribution functions to their one-dimensional marginal distribution functions and as distribution functions whose one-dimensional margins are uniform.

Definition 4 *A function C is called a $Copula$ if only if is defined:*

$$C:[0,1]^d \to [0,1]$$

It has the following characteristics:

$C(u_1,\cdots,u_d) = 0$ If one of its components u_i is equal to zero.

$C(1,\cdots,1,u_i,1,\cdots,1) = u_i$

In addition, C must be $d-increasing$. Example, for $d = 2$, we have:

$C(u,v):[0,1]^2 \to [0,1]$

For any $0 \le u \le 1$ and $0 \le v \le 1$ we have the three following conditions:

$C(0,v) = C(u,0) = 0$

$C(1,v) = v$

$C(u,1) = u$

For any u and v , we define the $2-increasing$ propriety as:

$$C(u_1,v_1) - C(u_1,v_2) - C(u_2,v_1) + C(u_2,v_2) \ge 0$$

Definition 5 *According to $Sklar's$ theorem, if C is a Copula, and if $F_1,\cdots,F_d$ are a cumulative distribution functions (univariate), then:*

$$F(x_1,\cdots,x_d) = C(F_1(x_1),\cdots,F_d(x_d))$$

is a cumulative distribution function with a dimension d , where the marginals are $F_1,\cdots,F_d$ exactly.

The converse is also true: if F is cumulative distribution function with d dimension, there is a C Copula such as:

$$F(x_1,\cdots,x_d) = C(F_1(x_1),\cdots,F_d(x_d))$$

where all F_i are F marginals' laws.

According to Sklar's theorem, two steps are performed in order to construct the joint probability distribution function of a random vector. The first step is constructing the margins of each random variable separately. The second step is selecting a proper Copula to construct the joint distribution. Therefore, Copulas can be used to study the distribution character of each random variable and their relationship.

There are many families of Copulas. They can be characterized by one parameter or by a vector of parameters. These parameters measure the dependence between the marginals and are called dependence parameters θ. This paper, use Frank Copula, a variant of Archimedean Copulas, because we obtained satisfactory results with it (see Section 6.1.3.1). In general, Archimedean Copulas have one dependence parameter θ that can be calculated using Kendall's τ [4].

Kendall's τ measures the concordance between two continuous random variables X1 and X2. The relation between Kendall's τ and θ in Frank Copula used in this paper is defined as:

$$\tau = 1 - \frac{4}{\theta}[1 - D_1(\theta)] \text{ where } D_1(\theta) = \frac{1}{\theta}\int_0^{\theta} \frac{t}{e^t - 1} dt$$

The Frank Copula function is defined by:

$$C(u, v; \theta) = -\frac{1}{\theta}\ln\left(1 + \frac{(e^{-\theta u} - 1)(e^{-\theta v} - 1)}{e^{-\theta} - 1}\right) \text{where } \theta \in (-\infty, \infty)$$

The dependence parameter of a bivariate Copula can be estimated using the maximum likelihood method (MLE). To do so, we need to optimize the log-likelihood function given by:

$$l(\theta) = \sum_{t=1}^{T} \ln c(F(x_{1t}), F(x_{2t}); \theta)$$

where T is the sample size. The value θ which maximizes the log-likelihood l(θ) is called maximum likelihood estimator $\hat{\theta}_{MLE}$. Once the value of θ is estimated, the bivariate Copula is well defined. For maximizing the likelihood function, we use the nonparametric estimation of θ given by Kendall's τ as an initial approximation to $\hat{\theta}_{MLE}$.

After the characterization of the Copula, the generation of sample is performed as the following steps:

1) Generate two independent uniform (0,1) variables u and t;

2) Set $v = C_u^{(-1)}(t)$, *where* $C_u^{(-1)}(t)$ *denotes a quasi-inverse of* $C_u(v)$.

3) The desired pair is (u, v).

4) $(x1, x2)$ *is a sample of the specified joint distribution, where* $x_1 = F_1^{(-1)}(u), x_2 = F_2^{(-1)}(v)$

VI. CEDA: Copula-Based Estimation of Distribution Algorithm

The aim of the proposal is to help the decision maker to get the solutions that are closest to its interest. To achieve this, a two-stage algorithm is proposed, that is composed of the *Optimization* stage which finds a set of the best solutions to a given problem (see Section 6.1) and the *Update* stage which finds another set of the best solutions until the decision maker is satisfied (see Section 6.2). Note that the Update stage runs much faster in finding new solutions compared to the initial Optimization stage.

A. Optimization Stage

Like every evolutionary algorithm the proposed algorithm (Algorithm 2) has two principal steps (i) the *Selection* and (ii) the *Reproduction*. In the Selection step (performed by Function SelectUsingMOEA), the proposal use the *NSGA2* [5] or *SPEA2* [6] to select the best individuals (solutions) that will be used in the Reproduction step where CEDA makes use of Copulas to estimate and regenerate new individuals (performed by Functions ConstructCopulas and GenerateSolutions respectively).

A pseudo-code of the algorithm that performs the estimation of distribution using a Copula for solving multiobjective problems can be viewed as follows (Algorithm 2).

Algorithm 2 Copula-based EDA

Function CEDA

P_0 = Initialization(m)

P = SelectUsingMOEA(P_0)

While Not termination criteria do

C = ConstructCopulas(P)

P' = GenerateSolutions(C)

P" = SelectUsingMOEA($[P'P]^T$)

P = P"

End while

Return (P, C)

End function

1) Initialization

Initially CEDA assume that we have a population $\mathbf{P_0} = [\mathbf{x_1}, \dots, \mathbf{x_m}]^T$ where $\mathbf{x_i}, i \in [1, m]$ are the individuals. Each individual $\mathbf{x_i} = [x_{i1}, \dots, x_{in}]$ where $x_{min} \leq x_{ij} \leq x_{max}$. Both x_{min} and x_{max} are reals. Where each x_{ij} is initially picked up according to a uniform distribution in $[x_{min}, x_{max}]$. Note that each individual $\mathbf{x_i}$, ($i \in [1, m]$) is real-value coded vector, i.e. every x_{ij}, ($i \in [1, m], j \in [1, n]$) are real values.

2) Selection

In selection step achieved by the function SelectUsingMOEA, CEDA use one of the classical algorithms NSGA2 or SPEA2 as a MOEA.

The result of the selection is a set of individuals that will be used in the reproduction step. The proposal call **P** the matrix of the individuals resulting from the selection process operated on the precedent population. For the first generation, the algorithm use the initial population $\mathbf{P_0}$. **P** is defined as the following:

$$\mathbf{P} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}$$

NSGA2 or SPEA2 selects the best individuals of the population it operates on according to the dominance relation defined in Section 2. Note that generated solutions (obtained by GenerateSolutions) at a given step are not necessarily better than those generated at the step that preceded it. Therefore, the selection of the best solutions operates on the union of the two sets: the solutions obtained from the current step and those resulted from the step that preceded it, as shown in Algorithm 2.

3) Reproduction

To perform the reproduction, CEDA start by calculating the dependency between the best individuals using Copula as shown in Algorithm 3 and then generate new individuals using these Copulas as shown in Algorithm 4.

a) Constructing Copulas

The Copula type used in this paper is Archimedean. The Archimedean Copula deals with two vectors of variables **u** and **v**; therefore, CEDA divided each one of the decision variable vectors into two sub vectors to fit into the variables **u** and **v**. CEDA performed this division into two sub vectors in each generation of the algorithm.

$$\mathbf{P} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} = [\mathbf{w_1}, \dots, \mathbf{w_n}]$$

CEDA operate on the transpose of the matrix **P,** and take each vector $\mathbf{w_j} = (x_{1j}, x_{2j}, \dots, x_{mj})^T$ the proposal take each vector $\mathbf{w_j}(j \in [1, n])$ to construct the sub vectors $\mathbf{u_1}, \dots, \mathbf{u_n}, \mathbf{v_1}, \dots, \mathbf{v_n}$ according to the following: $\mathbf{u_1}, \mathbf{v_1}$ are extracted from $\mathbf{w_1}$ where the size of each of $\mathbf{u_1}$ and $\mathbf{v_1}$ are equal to $m/2$. The elements of $\mathbf{u_1}$ are taken randomly from $\mathbf{w_1}$, and $\mathbf{v_1}$ is constructed from the rest of the elements of $\mathbf{w_1}$. For the sake of simplicity, CEDA assume that m is an even number. In the case where m is odd, CEDA remove one individual from the initial population to make its size even. The computation of the other sub vectors $\mathbf{u_2}, \dots, \mathbf{u_n}$ and $\mathbf{v_2}, \dots, \mathbf{v_n}$ is performed in a similar way as for $\mathbf{u_1}$ and $\mathbf{v_1}$ respectively.

The algorithm create Archimedean Copulas, represented by the vector $C = [C_1 \dots C_j \dots C_n]$, using the sub vectors $\mathbf{u_1}, \dots, \mathbf{u_n}, \mathbf{v_1}, \dots, \mathbf{v_n}$, where each Copula $C_j, j \in [1, n]$ is constructed from the sub vectors $\mathbf{u_j}$ and $\mathbf{v_j}$ as shown in Algorithm 3.

Algorithm 3 Construct Copulas

Function ConstructCopulas(P,type)

For all w_j a vector in P do

u_j = RandomPick(w_j)

v_j = Remainder(w_j,u_j)

C_j = Copula(u_j, v_j, type)

End for

Return C = [C_1 … C_j … C_n]

End function

Note that there are many types of Archimedean Copulas. In this paper, CEDA considered three of them namely Gumbel, Clayton and Frank Copula. CEDA have experimented with them on various optimization problems and found that Frank Copulas provides better results in the configurations tested on.

b) Generating New Individuals

The proposal uses the constructed Copulas $C_1, \dots, C_n$ to generate new individuals. The set of the new generated individuals $\mathbf{X'} = [\mathbf{w'_1}, \dots, \mathbf{w'_n}]$ where $\mathbf{w'_j}$ is the concatenation of $\mathbf{u'_j}$ and $\mathbf{v'_j}$ which are sampled using Copula C_j. Note that the vector $\mathbf{w'_j}$ (resulting from the concatenation of $\mathbf{u'_j}$ and $\mathbf{v'_j}$) is of size m' that is not necessarily the same of the initial population size m. The new individuals are therefore the vectors $\mathbf{x'_i}, i \in [1, m']$ where $\mathbf{X'} = [\mathbf{x'_1}, \dots, \mathbf{x'_{m'}}]^T$. Algorithm 4 summarizes theses steps.

Algorithm 4 Generate Solutions

Function GenerateSolutions(C, m')

For all Cj in C do

(u_j', v_j') = GenerateFromCopula(C_j, m')

w'$_j$ = Concat(u'$_j$, v'$_j$)

End for

return X' = [w'$_1$ … w'$_j$ … w$_n$']

End function

The function used to generate individuals form the estimated Copula C is performed in the same way defined in Section **5**. CEDA start by picking u and t form (0,1) uniform function then we get v by calculating the $C_u^{(-1)}(t)$ the quasi-inverse function of C_u. The generated variables x_1 and x_2 are produced from the quasi-inverse function of each marginal distribution. In every iteration (see Algorithm 4.a), CEDA insert x1 to the list **ListX1** and x2 to **ListX2**. Finally, after generating m samples of x1 and x2 we return the two lists.

Algorithm 4.a **GenerateFromCopula**

Function GenerateFromCopula(C, m)

For i=1, m do

$\mathbf{u} = \mathbf{uniform}(\mathbf{0}, \mathbf{1})$;

$\mathbf{t} = \mathbf{uniform}(\mathbf{0}, \mathbf{1})$;

$\mathbf{v} = \mathbf{C_u^{(-1)}}(\mathbf{t})$;

$\mathbf{x_1} = \mathbf{F_1^{(-1)}}(\mathbf{u})$;

$\mathbf{x_2} = \mathbf{F_2^{(-1)}}(\mathbf{v})$;

Insert (x1,ListX1)

Insert (x2,ListX2)

End for

Return (**ListX1**, **LisTX2**)

End function

B. Update Stage

The Optimization stage allows us to calculate new solutions as shown in Algorithm 2. These solutions may not suit the needs of the decision maker and thus another stage of new solutions generation is needed. The Update stage that proposed in this paper (as shown in Algorithm 5) makes it possible for the decision maker to find other new solutions quickly by using the Copulas constructed in the Optimization stage. Specifically, CEDA achieves this by calling Function *GenerateSolutions* with arguments C (the Copulas constructed in the Optimization

phase), and m″ the number of new individuals required. The output of the Update stage is the population P_{update}. If the decision maker is still not satisfied with the obtained solutions, only another round of the Update stage is required thus saving the need for running the Optimization stage another time.

Algorithm 5 Update Solutions

Function UpdateSolutions(C,m")

P_{tmp} = GenerateSolutions(C,m")

P_{update} = SelectUsingMOEA(P_{tmp})

Return P_{update}

End function

It is important to note that Copulas used as input in the Update Solution Algorithm have been constructed using a set of the best solutions obtained in the last generation of the algorithm used in the Optimization stage. Therefore, those Copulas inherently characterize the distribution of the best solutions thereby making the new individuals P_{tmp} among the best solutions. The returned solutions at this stage (Update stage) P_{update} are selected from the temporary individuals P_{tmp} according to one of the MOEA to select the best solutions as shown in Algorithm 5.

VII. Experimentation

A. Used Benchmark problems

To evaluate the efficiency of the proposed algorithm, we chose to test it on a set of benchmark problems usually used in the literature. Specifically, CEDA uses the benchmark problems UF1, UF2, …, UF10, CF1, CF2, …, and CF3 defined in CEC2009 competition [35]. CEDA operates on 100 individuals and set the maximum number of evaluation to 300000. Each algorithm runs independently 30 times for each benchmark problem, as recommended by CEC2009 settings. We vary the number of DM calls and show the results obtained with 5 and 20 DM calls.

B. Used Metrics

In addition to considering the metrics traditionally used to assess the quality of the obtained solutions, namely diversity and convergence, the proposal defines a new metric, **directed regeneration speed**, to measure how quickly new solutions can be obtained. The new metric allows showing the efficiency of the algorithm that enables finding new solutions according to the decision maker needs quickly without compromising their qualities.

Both the diversity and the convergence are calculated from the set of solutions obtained by the used algorithms (CEDA, SPEA2, NSGA2). The diversity of a set of solutions is calculated using the IGD metric defined in [10] to assess the quality of the distribution of the obtained solutions over the PF and the convergence to the PF.

The **solution updating speed** metric, referred to as I_{new}, measures the number of new solutions obtained over a period of time (expressed in terms of the number of function evaluations) as shown in (3).

To show the new aspect guaranteed with the algorithm, which is the ability to get new PS with a very short time (negligible) we have proposed a metric that calculate the number of different PS between two set of PS that can be defined as follows:

$$I_{new} = \frac{\sum_{t=0}^{T} |PS_t|}{\sum_{t=0}^{T} FE_t} \quad (3)$$

where $|PS_t|$ represents the number of Pareto Solutions obtained at iteration t, $|FE_t|$ is the number of the function evaluations, and T is the number of iterations (the number of times the decision maker calls the algorithm again to find new solutions).

C. Simulation Results

This section, shows that the proposed CEDA method achieves good diversity and convergence compared to those obtained with state-of-the-art methods such as SPEA2 and NSGA2 by considering benchmark problems (UF1, ..., UF10 and CF1, …, CF10) taken from CEC2009 [35].

1) Solution Qualities

Figure 1, shows that the Pareto Front solutions obtained when solving the considered benchmark problems with the proposed method CEDA-SPEA2 (CEDA using SPEA2 as selection method) and CEDA-NSGA2 (CEDA using NSGA2 as selection method). We show that both CEDA algorithms find solutions with similar qualities independently of the algorithm used for the selection (SPEA2 or NSGA2), because solutions are generated according to the same Copulas-based technique — SPEA2 or NSGA2 are only used for the selection.

Figure 2 shows that CEDA-NSGA2 and CEDA-SPEA2 provide solutions with different qualities on benchmarks UF2 and UF1 because they use different techniques to find new solutions. Figure 1 and Figure 2 also show that the proposal always provides solutions that are close to the optimal Pareto Front, particularly on benchmarks UF7, UF4, and CF1.

Figure 3 shows that the proposed CEDA algorithm generates more Pareto Solutions compared to those obtained by traditional algorithm NSGA2. Similar results have been obtained with CEDA compared to traditional SPEA2.

We show that both variants of CEDA converge to the optimal Pareto Front in a way that is similar to NSGA2 and SPEA2 (see Figure 1, 2, and 3). This shows that the Copula estimator is very good and comparable to the classical genetic operators (mutation and crossover) in terms of reproduction.

2) Solution Convergence and Diversity

To evaluate the convergence and the diversity of CEDA during, the measure the IGD Indicator obtained with both CEDA variants (CEDA-NSGA2 and CEDA-SPEA2) as well as those obtained with NSGA2 and SPEA2.

Figure 4 and Figure 5 show that CEDA-NSGA2 (resp. CEDA-SPEA2) algorithm achieves mean IGD values that are

close to those obtained with NSGA2 (resp. SPEA2), with both 5 DM and 20 calls. Those IGD values reflect the good convergence and diversity qualities achieved by the method.

3) Solution Update Speed (Update Stage)

Figure 6 shows the IGD of the Pareto Front using the CEDA-NSGA2 and CEDA-SPEA2 in function of the number of function evaluations. Lower IGD values reflect better solution qualities. We show that the speed of generating better solutions achieved by the CEDA algorithms is higher than that achieved by traditional algorithms. For example, to decrease the value of IGD of the approximated Pareto Solutions of the UF4 problem using CEDA-NSGA2, from the beginning of the execution of the algorithm to 0.8, needs 500 function evaluations compared to 10000 required by NSGA2.

For example, in the plot of CEDA-NSGA2 in the subfigure corresponding to the UF1 benchmark, the results corresponding to one call are represented in the leftmost point. That point was found after running 1000 evaluations (as represented in the x-axis). The point next to it on the right corresponds to two calls, which was found after running 2000 evaluations (see corresponding value on the x-axis). The point next to the second point represents the results obtained for three calls, and so on until we reach the rightmost point, which corresponds to the results obtained for 20 calls. The same explanation applies to CEDA-SPEA2, as well as NSGA2 and SPEA2 in the other subfigures.

Note that counting the number of function evaluations in the optimization stage starts from the beginning of finding of new solutions until their convergence to the optimal Pareto Front. However, during the update stage, the CEDA algorithms generate new solutions, which assess the quality by calculating the IGD and compare it with the IGD found in the first phase. If the IGD of the update stage is smaller or equal to the one of the optimization stage which consider that the update stage converged too, which gives DM good alternative solutions.

4) New Solution Count (Update Stage)

Figure 4 and Figure 5, plots the number of new solutions obtained when a Decision Maker wants to generate new ones. CEDA tested the solution with two cases. In the first case, the Decision Maker makes 5 calls to Algorithm 5, and in the second one the Decision Maker 20 calls to the same Algorithm. By considering these two cases, the work aim to reflect various decision making needs requiring different numbers of algorithm calls to obtain Decision Maker satisfaction. We plotted the mean value of the number of new solutions averaged over a 30 simulation runs, as well as the standard deviation, maximum and minimum values. The work shows that CEDA-based algorithms, both CEDA-SPEA2 and CEDA-NSGA2, generate a significantly greater number of new solutions per objective function evaluation (i.e. new solutions are obtained with fewer objective function evaluations), compared to traditional SPEA2 and NSGA2 algorithms. This is because Copulas based techniques reduce the search space which becomes closer to the optimal Pareto Front thereby making it easier to find new solutions with a smaller number of objective function evaluations compared to SPEA2 and NSGA2.

VIII. CONCLUSIONS

CEDA was presented, a Copula-based Estimation of Distribution Algorithm, to improve the efficiency of solving multiobjective optimization problems. CEDA is based on the statistical properties of Copulas to estimate the distribution of a population and thus its ability to generate new individuals with similar properties. This feature makes CEDA particularly designed to promptly help a Decision Maker find alternative solutions to a multiobjective problem when the solutions obtained by traditional algorithms such as SPEA2 and NSGA2 do not satisfy his/her needs. The production of alternative solutions by CEDA is accelerated by the fact that they are generated according the probabilistic model established by the use of Copulas thereby saving the need for running the costly traditional MOEA algorithms another time to find new solutions.

CEDA was tested on a set of benchmark problems from the CEC2009[35] traditionally used by the community for the evaluation of multiobjective problem solving algorithms and shown that the proposal provides solutions with good convergence and diversity compared to state-of-the-art algorithms such as SPEA2 and NSGA2. This work have also particularly shown that the time needed to generate these solutions is substantially smaller than that needed by state-of-the-art algorithms, which makes the algorithm suitable for prompt alternative solution generation.

CEDA was tested with traditional NSGA2 and SPEA2 as the selection methods. Although the results are encouraging, better results with other methods such as the MOEA/D or a hybrid evolutionary algorithm [36-39] as substitutes for NSGA2 and SPEA2 is expected. CEDA Algorithms may also be used for solving MaOPs (Many-Objective Problems) where the there are more than four objectives to optimize. In addition, Copulas creations are independent and thus can be done in parallel, which will even enhance the performance on parallel computers.

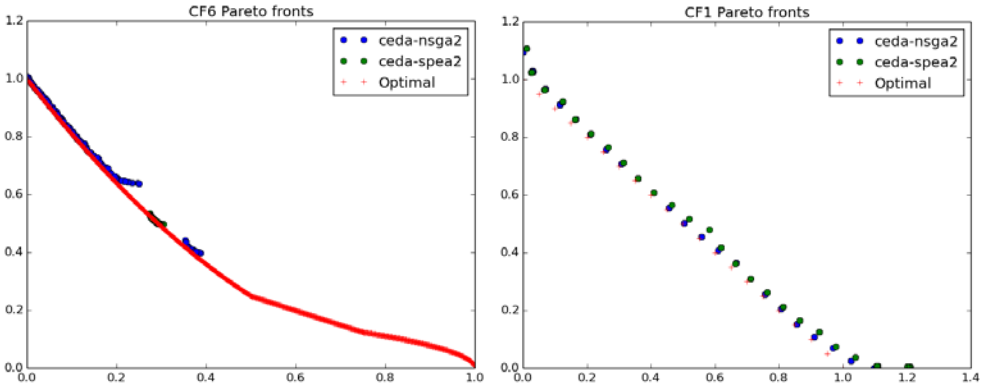

Fig. 1. Pareto front of the CF1 and CF6 Constrained problems

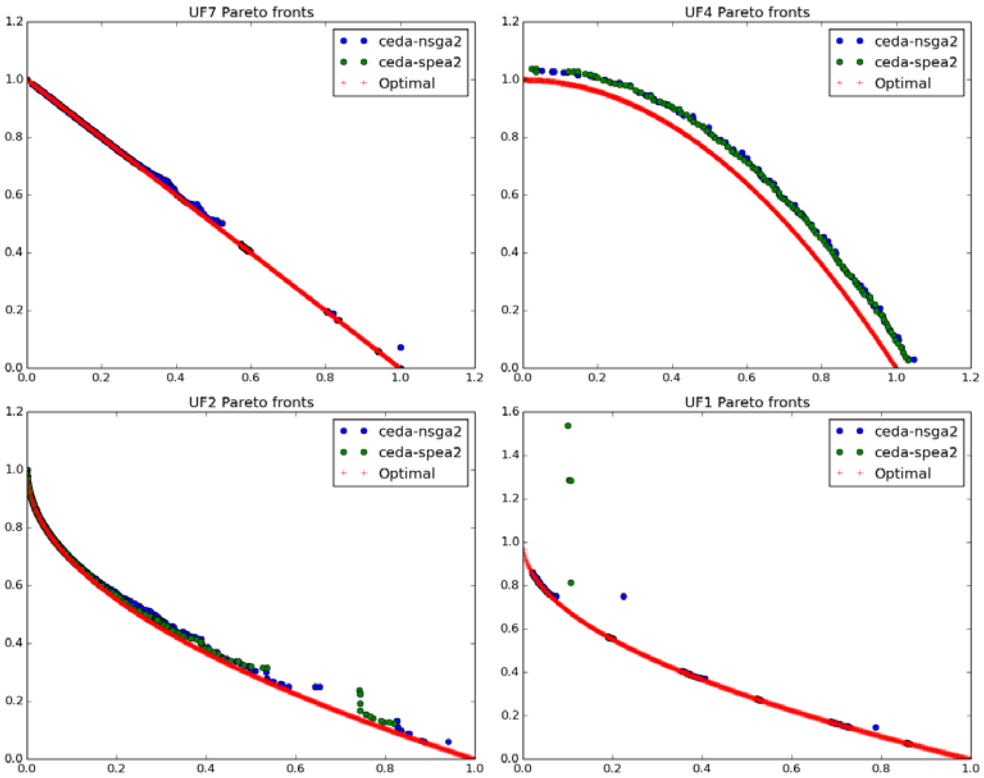

Fig. 2. Pareto front of the UF1,UF2,UF4 and UF7 unconstrained problems

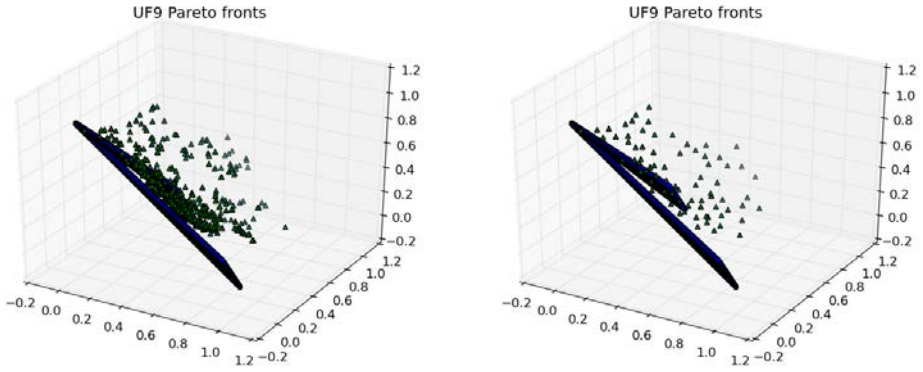

Fig. 3. Pareto Front of the CEDA-NSGA2 and NSGA2 of the UF9 problem

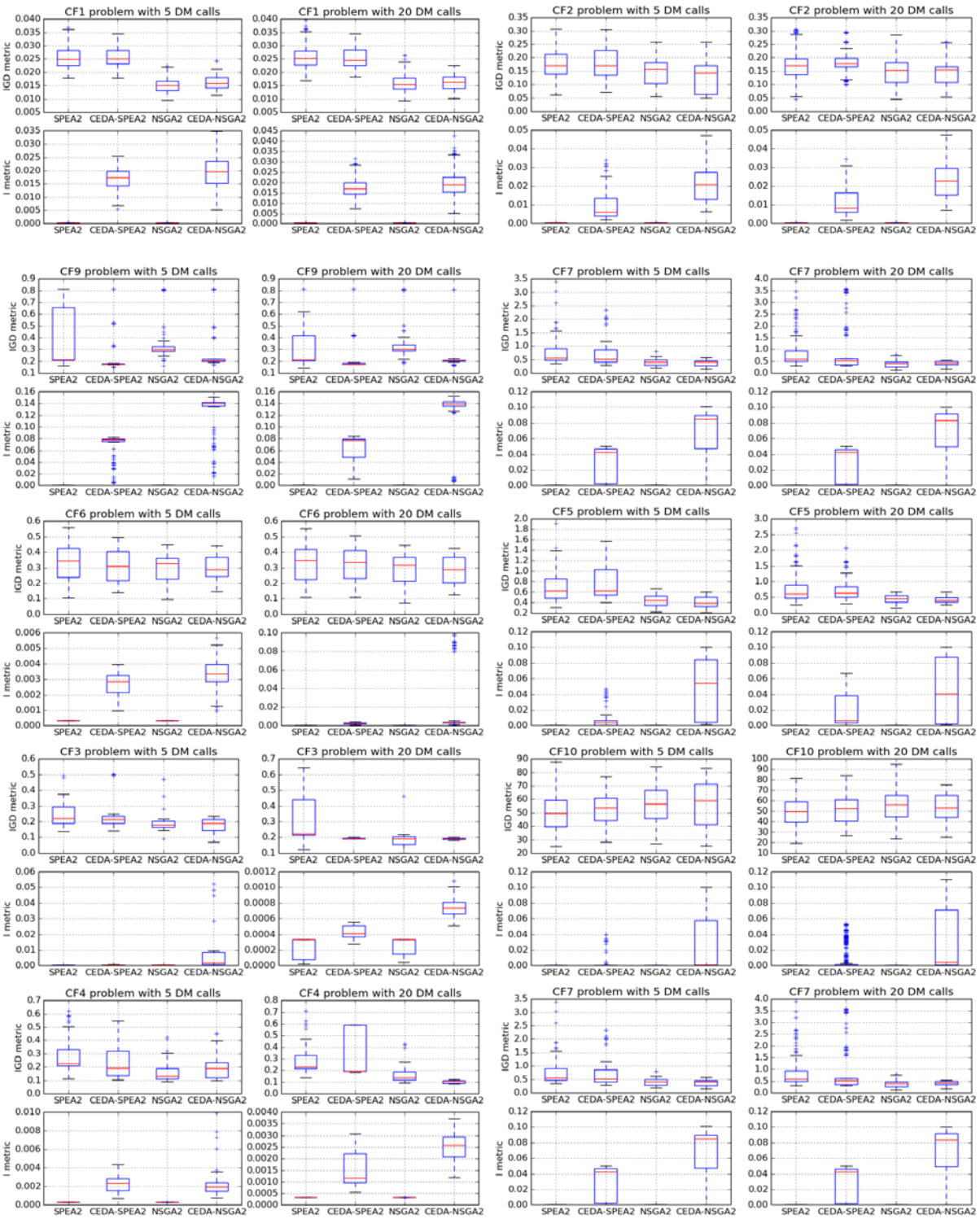

Fig. 4. Mean/Deviation of the *IGD* and I_{new} metrics for the constrained problems benchmarks

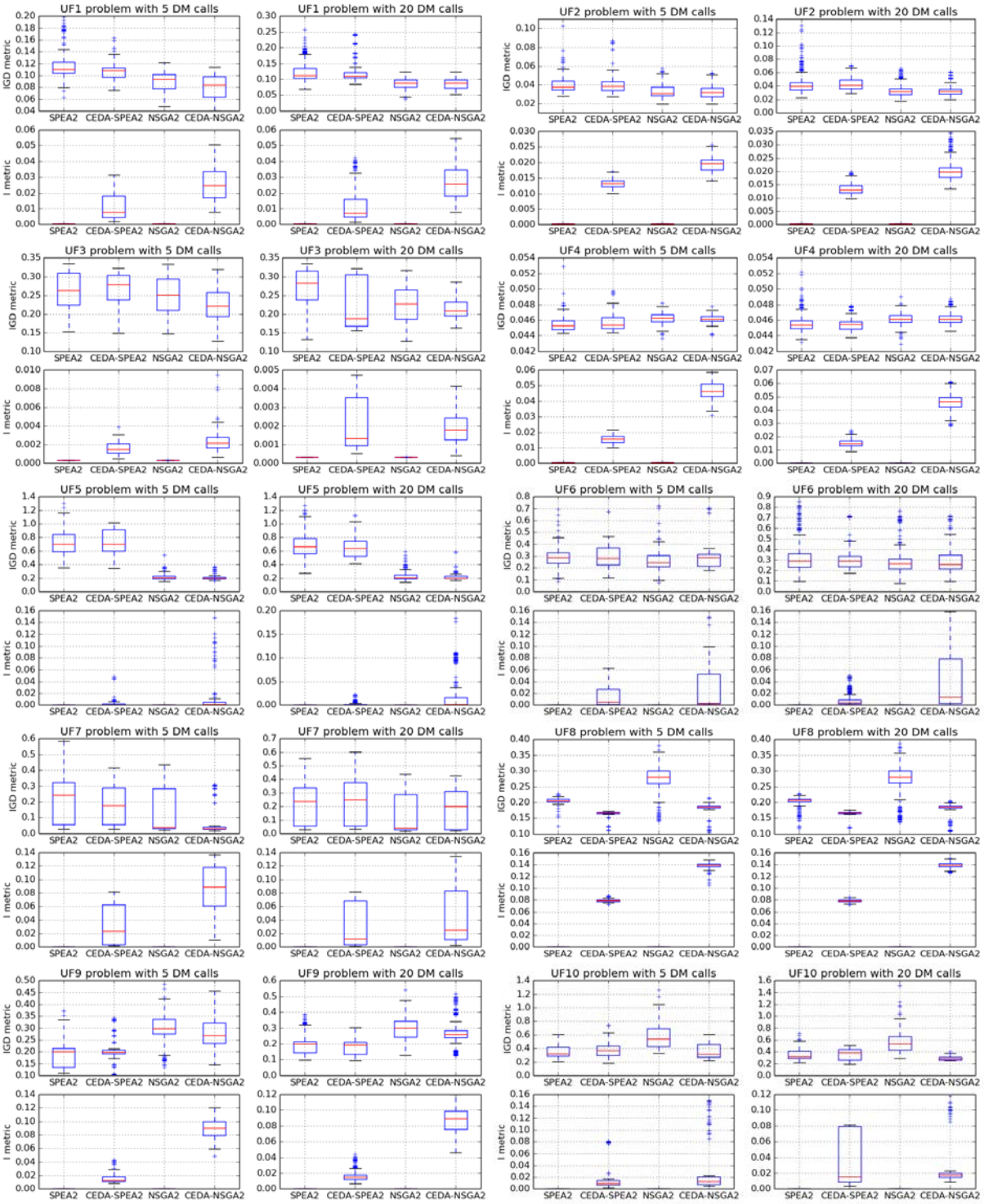

Fig. 5. Mean/Deviation of the *IGD* and I_{new} metrics for the unconstrained problems benchmarks

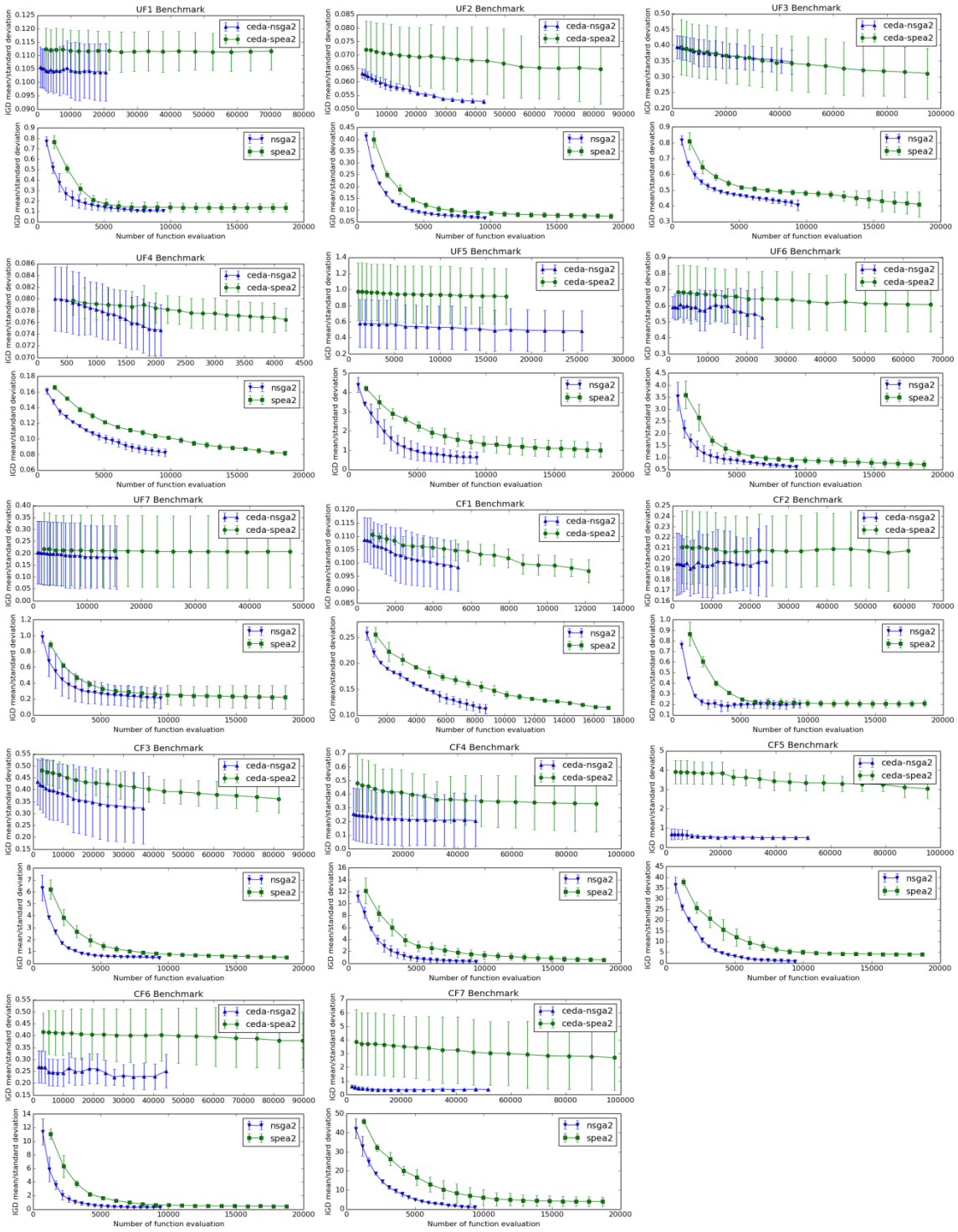

Fig. 6. The evolution of the means/standard deviations of IGD values of the approximate solution sets obtained with the number of function evaluations for the test instances

REFERENCES

[1] Deb K. Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, 2001.

[2] Hauschild M, Pelikan M. An introduction and survey of estimation of distribution algorithms. Swarm Evol Comput 2011; 1: 111–128.

[3] Auger A, Doerr B. Theory of Randomized Search Heuristics: Foundations and Recent Developments. vol. 1. World Scientific; 2011.

[4] Nelsen RB. An introduction to Copulas. Springer; 2006.

[5] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002; 6: 182–197.

[6] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Evolutionary Methods for Design, Optimisation, and Control; 2002; Barcelona, Spain. pp. 95–100.

[7] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results. Evol Comput 2000; 8: 173–195.

[8] Fonseca CM, Fleming PJ. Multiobjective optimization and multiple constraint handling with evolutionary algorithms II Application example. IEEE Trans Syst Man Cybern Part A Syst Humans 1998; 28: 38–47.

[9] Zhou A, Qu BY, Li H, Zhao SZ, Suganthan PN, Zhang Q. Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm Evol Comput 2011; 1: 32–49.

[10] Zhang Q, Li H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Trans Evol Comput 2007; 11: 712–731.

[11] Zitzler E, Simon K. Indicator-Based Selection in Multiobjective Search. In: Yao X, Burke EK, Lozano J, Smith J, Merelo-Guervos JJ, Bullinaria JA, Rowe JE, Tino P, Kabàn A, Schwefel HP,editors. 8th International Conference on Parallel Problem Solving from Nature.Berlin, Heidelberg: Springer, 2004. pp. 832–842.

[12] Brockhoff D, Zitzler E. Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In: IEEE Congress on Evolutionary Computation; Sept 2007; Singapore. IEEE. pp. 2086–2093.

[13] Bader J, Zitzler E. HypE: an algorithm for fast hypervolume-based many-objective optimization. Lect Notes Comput Sc 2011; 19: 45–76.

[14] Branke J, Deb K. Integrating User Preferences into Evolutionary Multi-Objective Optimization. In: Jin Y, editor. Knowledge Incorporation in Evolutionary Computation. vol. 167.Berlin, Heidelberg: Springer, 2005. pp. 461–477.

[15] Deb K, Jain H. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. IEEE Trans Evol Comput 2014; 18: 577–601.

[16] Sinha A, Korhonen P, Wallenius J, Deb K. An interactive evolutionary multi-objective optimization algorithm with a limited number of decision maker calls. Eur J Oper Res 2014; 233: 674–688.

[17] Fleetwood K. An Introduction to Differential Evolution, New ideas in optimization. UK, Maidenhead, UK: McGraw-Hill Ltd, 1999.

[18] Coello CAC, Cortés NC. Solving multiobjective optimization problems using an artificial immune system. Genet Progam Evolvable Mach 2005; 6: 163–190.

[19] Reyes-Sierra M, Coello CC. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International journal of computational intelligence research 2006; 2: 287–308.

[20] Ünveren A, Acan A, editors. Multi-objective optimization with cross entropy method: Stochastic learning with clustered pareto fronts. In: IEEE Congress on Evolutionary Computation; 25-28 Sept. 2007; Singapore. IEEE. pp. 3065 – 3071.

[21] [21] Han KH, Kim JH. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans Evol Comput 2002; 6: 580–593.

[22] Muhlenbein H, Paaß G. From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In: Voigt HM, Ebeling W, Rechenberg I, Schwefel HP, editors. Parallel Problem Solving from Nature PPSN IV. Springer Berlin Heidelberg, 1996. pp. 178–187.

[23] Salinas-Gutierrez R, Hernandez-Aguirre A, Villa-Diharce ER. Estimation of Distribution Algorithms Based on Copula Functions. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation; 12-16 July 2011; New York, NY, USA: ACM. pp. 795–798.

[24] Wang L, Wang Y, Zeng J, Hong Y. An estimation of distribution algorithm based on Clayton copula and empirical margins. In: Li K ,Li X ,Ma S ,Irwin GW, editors .Life System Modeling and Intelligent Computing. Berlin, Heidelberg: Springer, 2010. pp. 82–88.

[25] Salinas-Gutiérrez R, Hernàndez-Aguirre A, Villa-Diharce ER. Using copulas in estimation of distribution algorithms. In: Aguirre AH, Borja RM, Garcià CAR, editors. MICAI 2009:Advances in Artificial Intelligence. Berlin, Heidelberg: Springer, 2009. pp. 658–668.

[26] Gao Y, Peng L, Li F, Liu M, Hu X. EDA-Based Multi-objective Optimization Using Preference Order Ranking and Multivariate Gaussian Copula. In: Guo C, Hou ZG, Zeng Z, editors. Advances in Neural Networks. Berlin, Heidelberg: Springer, 2013. pp. 341–350.

[27] Gao Y, Peng L, Li F, Liu M, Hu X. Multiobjective Estimation of Distribution Algorithms Using Multivariate Archimedean Copulas and Average Ranking. In: Wen Z, Li T, editors. Foundations of Intelligent Systems. Berlin, Heidelberg Springer, 2014. pp. 591–601.

[28] Wang LF, Zeng JC, Hong Y. Estimation of distribution algorithm based on archimedean copulas. In: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation. ACM; June 12-14 2009 ; Shanghai, China. New York, NY, USA :ACM. pp. 993–996.

[29] Salinas-Gutiérrez R, Hernàndez-Aguirre A, Villa-Diharce ER. D-vine EDA: a new estimation of distribution algorithm based on regular vines. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation; 7-11 July 2010; Portland, Oregon. New York, NY, USA :ACM. pp. 359–366.

[30] Gao Y. Multivariate estimation of distribution algorithm with laplace transform archimedean copula. In: Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on. IEEE; 2009. pp. 1–5.

[31] Wang L, Wang Y, Zeng J, Hong Y. An estimation of distribution algorithm based on clayton copula and empirical margins. In: Life System Modeling and Intelligent Computing. Springer; 2010. pp. 82–88.

[32] Wang X, Gao H, Zeng J. Estimation of Distribution Algorithms Based on Two Copula Selection Methods. Int J Comput Sci Math. 2012 Jan; 3: 317–331.

[33] Chang C, Wang L. A multi-population parallel estimation of distribution algorithms based on Clayton and Gumbel copulas. In: Deng H, Miao D, Lei J, Wang FL, editors. Artificial Intelligence and Computational Intelligence .Berlin, Heidelberg: Springer, 2011. pp. 634–643.

[34] Wang L, Guo X, Zeng J, Hong Y. Using gumbel copula and empirical marginal distribu- tion in estimation of distribution algorithm. In: Advanced Computational Intelligence (IWACI), 2010 Third International Workshop on; 25-27 Aug 2010; Suzhou, Jiangsu IEEE. 2010.pp.583–587.

[35] Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., and Tiwari, S.: 'Multiobjective optimization test instances for the CEC 2009 special session and competition', University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report, 2008, pp. 1-30

[36] Mashwani, W.K., and Salhi, A.: A decomposition-based hybrid multiobjective evolutionary algorithm with dynamic resource allocation, Applied Soft Computing, 2012, 12, (9), pp. 2765-2780

[37] Mashwani, W.K.: 'Hybrid Multiobjective Evolutionary Algorithms: A Survey of the Stateof-the-art', International Journal of Computer Science Issues (IJCSI), 2011, 8, (6), pp. 374-392

[38] Mashwani, W.K.: 'Comprehensive Survey of the Hybrid Evolutionary Algorithms', Int. J. Appl. Evol. Comput., 2013, 4, (2), pp. 1-19

[39] Mashwani, W.K., and Salhi, A.: 'Multiobjective memetic algorithm based on decomposition', Applied Soft Computing, 2014, 21, pp. 221-243