Evolutionary Intelligence
https://doi.org/10.1007/s12065-020-00523-4

RESEARCH PAPER

Vine copula-based EDA for dynamic multiobjective optimization

Abdelhakim Cheriet[1]

Received: 7 April 2020 / Revised: 22 October 2020 / Accepted: 30 October 2020

Abstract
Dynamic Multiobjective Problems cover a set of real-world problems that have many conflicting objectives. These problems are challenging and well known by the dynamic nature of their objective functions, constraint functions, and problem parameters which often change over time. In fact, dealing with these problems has not been investigated in detail using the Estimation of Distribution Algorithms (EDAs). Thus, we propose in this paper an EDA-based on Vine Copulas algorithm to deal with Dynamic Multiobjective Problems (DMOPs). Vines Copulas are graphical models that represent multivariate dependence using bivariate copulas. The proposed Copula-based Estimation of Distribution Algorithm, labeled Dynamic Vine-Copula Estimation of Distribution Algorithm (DynVC-EDA), is used to implement two search strategies. The first strategy is an algorithm that uses the model as a memory to save the status of the best solutions obtained during the current generation. The second strategy is a prediction-based algorithm that uses the history of the best solutions to predict a new population when a change occurs. The proposed algorithms are tested using a set of benchmarks provided with CEC2015 and the Gee-Tan-Abbass. Statistical findings show that the DynVC-EDA is competitive to the state-of-the-art methods in dealing with dynamic multiobjective optimization.

Keywords Evolutionary algorithm · Copula · Vine-copula · EDA · Dynamic multiobjective optimization

1 Introduction

In various fields of science and technology, optimization problems have two or more objectives that should be optimized simultaneously. These problems are called Multiobjective Optimization Problems (MOPs) [14], and their solution involves the design of algorithms that are different from those adopted for solving single-objective optimization problems.

In the absence of reference information, there is no unique or straightforward way to determine if a given solution is better than others in multiobjective optimization. The notion of optimality most commonly adopted in such context is the one called Pareto optimality. Pareto optimality leads to trade-offs among the objectives in MOPs. Thus, the solution of MOPs is usually a set of acceptable trade-off optimal solutions called Pareto optimal set. Compared to traditional algorithms, evolutionary algorithms (EAs) have shown good results in solving problems in many research areas, including engineering, biology, robotics [51], classification and clustering [2, 3], machine learning [5], information retrievals [4] and optimization. Namely, Non-dominated Sorting Genetic Algorithm (NSGA2) [18], Strength Pareto Evolutionary Algorithm (SPEA2) [68] and Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) [63] has been proposed and then widely used for solving MOPs in the last decade.

After demonstrating their usefulness in finding multiple Pareto-optimal solutions for static multiobjective optimization problems, there is a growing need for solving Dynamic Multiobjective Optimization Problems (DMOPs) using Evolutionary Algorithms [21]. A dynamic optimization problem involves objective functions, constraint functions, and problem parameters, that change with time [17]. This class of problems is more challenging than static MOPs since they combine the known difficulties of obtaining a diverse set of solutions with the dynamic changes that occur on the objective functions. That is, optimal solutions will vary, following the change of the objective function that varies over time.

✉ Abdelhakim Cheriet
Abdelhakim.cheriet@univ-ouargla.dz

[1] Department of Computer Science and Information Technologies, Kasdi Merbah University of Ouargla, Ouargla, Algeria

Published online: 16 November 2020

Recently, Estimation of Distribution Algorithms (EDAs) [37], a subclass of evolutionary algorithms, has seen an increasing number of applications to different kinds of optimization problems. EDAs can be seen as an entirely new paradigm of evolutionary computation, which is a combination of statistical learning theory and stochastic optimization algorithm, without the use of conventional evolutionary operators such as crossover and mutation. EDA works by estimating the probabilistic distribution of the selected population and describes its evolutionary trend directly from a macroscopic point of view. It has been shown that EDA exhibits some special characteristics of the concise concept and good global searching ability [37].

Hence, EDAs have been successfully extended to solve multiobjective optimization problems [39, 43, 47]. The performance of an EDA highly depends on how well it estimates and samples the probability distribution. A wide variety of EDAs using probabilistic graphical modeling techniques to estimate and sample the probability distribution have been proposed in the literature (e.g. [20, 38, 48]) and still the subject of active research. However, EDAs using probabilistic graphical modeling techniques may spend too much time on learning the probability distribution of the candidate solutions.

The essential component and distinguishing characteristic of an EDA is its probabilistic model. These probabilistic models depend on the representation of the problem and the type of interactions to capture. One of the practical probabilistic models that have been used in EDA is copula [44], especially in case of problems with continuous representations.

Copula was introduced by Sklar [50] as a tool for constructing multi-variate distributions and it was successfully used in many research [13, 41]. According to copula theory, a joint probability distribution can be decomposed into n marginal probability distributions and a copula function. A joint probability distribution can be constructed utilizing a copula function and the marginal probability distributions of every variable.

Multi-variate Archimedean copula [59] is a variant of the Archimedean copulas family, which has attracted particular interest due to their ability to model dependence in high dimensions with only one parameter. They have also been used as probabilistic models for EDAs [59]. However, learning high-dimensional copulas requires more complex estimation procedures that can also be more time-consuming compared to bi-variate models. To deal with this issue, the authors in [11, 12] have proposed to use a Vine copula model that is characterized by the positive effect of speeding up multiobjective optimization computation time.

One of the bi-variate models based on copulas that provide high flexibility for modeling multi-variate distributions is the vine copula. In simple terms, a vine copula is a set of nested trees where each edge in a tree represents a dependence relationship that is encoded with a copula function.

In this paper, we introduce the Dynamic Vine Copula-based Estimation of Distribution Algorithm (labeled DynVC-EDA), which learns a vine model with Archimedean copulas to solve dynamic multiobjective problems. The vine model is used to describe the distribution and dependencies between the variables of the obtained Pareto solutions.

In this proposition, we introduce two strategies labeled DynVC-EDA-M and DynVC-EDA-P. The first strategy (DynVC-EDA-M) is represented by an algorithm that uses the vine model as a memory to save the status of the best solutions obtained during the current generation. The second strategy (DynVC-EDA-P) is a prediction-based algorithm that uses the history of the best obtained solutions to predict a new population when a change occurs (i.e. an algorithm that predicts upcoming changes on the problem).

Convergence and diversity metrics are used to evaluate the performance of the proposed algorithms. We tested DynVC-EDA on a set of dynamic benchmark traditionally used by the community, namely the FDA benchmarks [21], CEC2015 benchmarks [30], and the new Gee-Tan-Abbass (GTA) benchmarks [24]. We used DynVC-EDA along with the well known multiobjective evolutionary algorithm based on decomposition (MOEA/D) [62] as candidates examples to select the best solution in every iteration. MOEA/D decomposes a multiobjective optimization problem into several single-objective optimization sub-problems and then optimize these sub-problems. MOEA/D uses many decomposition methods; in this paper, we choose to use Tchebycheff as a decomposition method [62].

The rest of the paper is organized as follows: Sect. 2 describes the background of the problem addressed in this proposal. We also provide some concepts and definitions related to the dynamic multiobjective problem. Then, we introduce some relevant concepts from the copula theory, which plays a primary role in this work. Section 3 explains the proposed dynamic multiobjective optimization methods. Both the memory-based algorithm, and the prediction-based algorithm are presented in details. Section 4 describes the design of experiments, introduces the benchmarks and the metrics used for evaluating this work. Section 5 discusses the results achieved by the proposed algorithms in comparison to other methods. Finally, we conclude the paper in Sect. 6.

2 Background

2.1 Dynamic multiobjective optimization

Most real-world problems are multiobjective and often they are dynamic and in continuous change over time. These changes affect the objective function, the problem instances,

and/or the objective constraints. In the literature, researchers usually define optimization problems that change over time as dynamic problems or time-dependent problems [42].

2.1.1 Problem definition

Definition 1 A dynamic multiobjective problem can be represented as the following multiobjective optimization problem [21]. Let t be the time variable, $\mathbf{V}$ and $\mathbf{W}$ be n-dimensional and m-dimensional continuous or discrete vector spaces, $\mathbf{g}$ and $\mathbf{h}$ be two functions defining inequalities and constraints, and $\mathbf{f}$ be a function from $\mathbf{V} \times t$ to $\mathbf{W}$. A dynamic multiobjective minimization problem with m objectives $f_1, \ldots, f_m$ is defined in Eq. (1):

$$\begin{aligned} &\min_{v \in V} f = (f_1(v,t), \ldots, f_m(v,t)) \\ &s.t \quad g(v,t) \leq 0 \text{ and } h(v,t) = 0 \end{aligned} \tag{1}$$

Definition 2 Considering a minimization problem, a decision vector $\mathbf{u}$ dominates $\mathbf{v}$ at time t ($\mathbf{u} \prec \mathbf{v}$) iff $f_i(\mathbf{u},t) < f_i(\mathbf{v},t), \forall i \in \{1, \ldots, m\}$.

Definition 3 The solution of a multiobjective problem is a set of solutions which are not dominated by any other solution at a specific time t. We call this set the **Pareto Optimal Solutions POS** at time t. The image of this set in the objective space forms the **Pareto Optimal Front POF** at time t.

2.1.2 Types of dynamic problems

Concerning the way POS and POF change, there are four possible ways a problem can demonstrate a time-varying change according to [21]:

1. The POS (optimal decision variables) changes, whereas the POF (optimal objective values) does not change.
2. Both POS and POF change.
3. POS does not change, whereas POF changes.
4. Both POS and POF do not change, although the problem can change.

2.2 Copula theory

Sklar [50] introduced the functions of the copula as a powerful tool to model the dependency between variables. Copula-based methods do not require all hypotheses about the probability distribution of the data. The method exploits the linear and non-linear relations between two or more variables using a copula function on an empirical bi-variate or multi-variate distribution. Finally, new data that are compatible with the previous derived dependency structure can be simulated with a cumulative distribution function given by this copula.

2.2.1 Sklar's theorem

Sklar theorem [50] indicates that a copula C attaches a given multi-variate distribution function to their univariate marginals. In a bi-variate distribution, and for the conjoint distribution $F_{XY}(x,y)$ with the univariate marginal distribution functions $F_X(x)$ and $F_Y(y)$, there exists a copula C.

$$F_{XY}(x,y) = C(F_X(x), F_Y(y)) = C(u,v) \tag{2}$$

$$C : [0,1]^2 \rightarrow [0,1] \tag{3}$$

If C is a copula and F_X and F_Y are distribution functions, F_{XY} can be defined as a conjoint distribution with F_X and F_Y. The probability density function (PDF) of the bi-variate distribution $f_{X,Y}(x,y)$ is represented in Eq. (4):

$$f_{XY}(x,y) = C(F_X(x), F_Y(y)) \cdot f_X(x) \cdot f_Y(y) \tag{4}$$

where $f_X(x)$ and $f_Y(y)$ are the univariate marginals of X and Y. The copula is unique when the marginals are continuous.

2.2.2 Copula characteristics

Some copula characteristics are given by Genest and Rivest [26] and resumed as follows. In the bi-variate case, a copula is represented as a function C from $[0,1]^2$ to $[0,1]$ with $\forall u, v \in [0,1]$:

$$C(u,0) = C(0,v) = 0 \tag{5}$$

$$C(u,1) = u \text{ and } C(1,v) = v \tag{6}$$

A copula is an increasing function $\forall u_1, u_2, v_1, v_2 \in [0,1]$ with $u_1 \leq u_2$ and $v_1 \leq v_2$ gives

$$C(u_2,v_2) - C(u_2,v_1) - C(u_1,v_2) - C(u_1,v_1) \geq 0 \tag{7}$$

A copula is a continuous function:

$$|C(u_2,v_2) - C(u_1,v_1)| \leq |u_2 - u_1| - |v_2 - v_1| \tag{8}$$

An important class of copulas are the Archimedean copulas [26]. This class is characterized by the ease with which they can be constructed and the nice properties they possess. Archimedean copulas are defined by:

$$C(u,v) = \psi^{-1}(\psi(u) + \psi(v)) \tag{9}$$

where ψ is the generator of the copula.

There are three Archimedean copulas in common use: the Clayton, Frank, and Gumbel (Fig. 1).

The Clayton copula is an asymmetric Archimedean copula, exhibiting greater dependence in the negative tail than in the positive. This copula is given by:

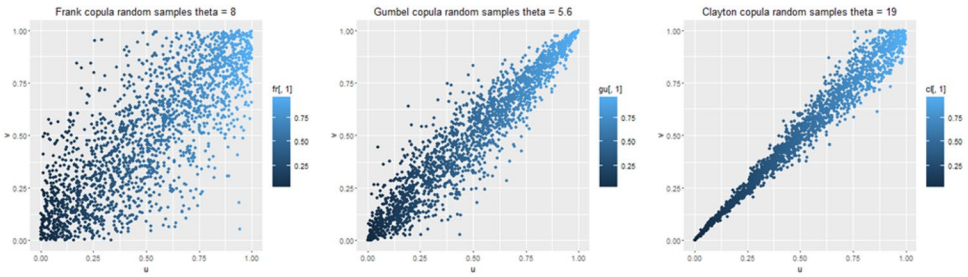


Fig. 1 Archimedean copulas samples

$$C_\theta(u,v) = \max\left(\left[u^{-\theta} + v^{-\theta} - 1\right]^{1/\theta}, 0\right) \tag{10}$$

And its generator is:

$$\psi_\theta(t) = \frac{1}{\theta}\left(t^{-\theta} - 1\right) \tag{11}$$

where $\theta \in [-1, \infty)\backslash\{0\}$. The relationship between Kendall's τ and the Clayton copula parameter θ is given by $\hat{\theta} = \frac{2\tau}{1-\tau}$

Frank copula is a symmetric Archimedean copula given by:

$$C_\theta(u,v) = -\frac{1}{\theta}\log\left[1 + \frac{(e^{-\theta u} - 1)(e^{-\theta v} - 1)}{e^{-\theta} - 1}\right] \tag{12}$$

And its generator is:

$$\psi_\theta(t) = -\log\left(\frac{\exp(-\theta t) - 1}{\exp(-\theta) - 1}\right) \tag{13}$$

where $\theta \in (-\infty, \infty)\backslash\{0\}$, the relationship between Kendall's τ and the Frank copula parameter θ is given by $\tau = 1 + \frac{4}{\theta}(D_1(\theta) - 1)$ where $D_1(\theta) = \frac{1}{\theta}\int_0^\theta \frac{t}{e^t - 1}dt$ is a Debye function of order one [26].

The Gumbel copula is an asymmetric Archimedean copula, exhibiting greater dependence on the positive tail than on the negative one. This copula is given by:

$$C_\theta(u,v) = \exp\left[-\left((-\log(u))^\theta + (-\log(v))^\theta\right)^{1/\theta}\right] \tag{14}$$

And its generator is:

$$\psi_\theta(t) = (-\log(t))^\theta \tag{15}$$

where $\theta \in [1, \infty)$. The relationship between Kendall's τ and the Gumbel copula parameter θ is given by $\hat{\theta} = \frac{1}{1-\tau}$

2.2.3 Copula simulation

Copulas have a direct application in the simulation of the dependent variables. We present the general procedure to simulate bi-variate dependents variables. First, we start by a short description of what is a pseudo-inverse of a distribution function.

Definition 4 Let F be a univariate distribution function. A pseudo-inverse of F is function [19] $F^{[-1]} : \mathbb{I} \to \mathbb{R}$ such that for $t \in \mathbb{I}$:

$$F^{[-1]}(t) = \inf\{x : F(x) \geq t\} = \sup\{x : F(x) \leq t\} \tag{16}$$

A general algorithm to generate new samples (x, y) of a pair of random variables (X, Y) with the marginals F_X, F_Y, joint distribution F_{XY}, and a copula C is defined as follows. Initially, we generate a pair of observations (u, v) of random variables (U, V) and compute the copula C. Then, using the inverse transformation method, we transform (u, v) to (x, y) using :

$$\begin{aligned} x &= F_X^{[-1]}(u) \\ y &= F_Y^{[-1]}(v) \end{aligned} \tag{17}$$

To generate the pair (u, v) we perform the following steps:

- Sample from independent random variables u, t uniformly in $\mathbb{I}$.
- Define $v = C^{[-1]}(t)$.

The generate pair is then (u, v).

2.2.4 Vine Copulas

Though copulas can represent a wide variety of dependence relationships between pairs of variables, their extension to represent multi-variate distributions is cumbersome. Multi-variate copulas are not sufficiently flexible to represent dependence structures for which all pairs of variables share the same type of dependence. Pair copula constructions [1, 16] are an effective alternative to build multi-variate dependence models using bi-variate copulas. They can be represented using a particular type of graphical model called vine graphical model or simply vine [8, 53].

A vine on n variables is a nested set of trees $T_1, \ldots, T_{n-1}$, where the edges of tree j are the nodes of the $j+1$-th tree, with $j = 1, \ldots, n-2$. Two special cases of vines are canonical vines (C-vines) and drawable vines (D-vines). In C-vines, for each tree T_j there is a unique node connected to $n - j$ edges. In D-vines, for each tree T_j, there is no node that is connected to more than two edges.

The C-vine density is given by Eq. (18):

$$f(x_1,\ldots,x_n)=\prod_{k=1}^{n} f(x_k)\prod_{j=1}^{n-1}\prod_{i=1}^{n-j} c_{j,j+i|i,\ldots,j-1}, \tag{18}$$

and the D-vine density is defined by Eq. 19:

$$f(x_1,\ldots,x_n)=\prod_{k=1}^{n} f(x_k)\prod_{j=1}^{n-1}\prod_{i=1}^{n-j} c_{i,i+j|i+1,\ldots,i+j-1}, \tag{19}$$

where j represents the trees and i denotes the edges.

The arguments of the pair-copulas in (18) and (19) are conditional distributions of the form $F(x|\mathbf{v})$ determined by the subscripts of the copula. In [46] it is shown that:

$$F(x|\mathbf{v})=\frac{\partial C_{x,v_j|\mathbf{v_{-j}}}(F(x|\mathbf{v_{-j}}),F(v_j|\mathbf{v_{-j}}))}{\partial F(v_j|\mathbf{v_{-j}})} \tag{20}$$

where $C_{x,v_j|\mathbf{v_{-j}}}$ is a bi-variate copula distribution function, $\mathbf{v}$ is an n-dimensional vector, v_j is one component of $\mathbf{v}$ and $\mathbf{v_{-j}}$ denotes the $\mathbf{v}$-vector excluding the j component.

When x and v are uniform the recursive evaluation of $F(x|\mathbf{v})$ can be expressed as:

$$h(x,v,\theta)=F(x|v)=\frac{\partial C_{xv}(x,v,\theta)}{\partial_v} \tag{21}$$

where h is a function defined to facilitate the computation of $F(x|v)$ and θ denotes the set of parameters for the copula of the joint distribution of x and v.

2.3 Related work

In this section, we present the state-of-the-art methods related to our topic. First, we overview the evolutionary based methods proposed for Dynamic Optimization Problems. Then, we present copula-based approaches for the Estimation of Distribution Algorithms.

2.3.1 Evolutionary algorithms for dynamic multiobjective optimization

There are two main approaches to deal with dynamic multiobjective problems. The first one approach regroups memory-based algorithms, and the second one regroups prediction-based algorithms.

The memory-based algorithms use memory to store information from past generations. The stored information will guide the search process in future generations or when a change occurs in the problem. The memory-based approach was successfully applied in many research papers. For instance, the Dynamic Competitive Cooperative CO-EA [27] used an external population as a memory. The memory is used with the addition population archive to store information related to Pareto Front (PF). The Adaptive Population Management-Based Dynamic NSGA2 [7] is based on a technique that measures the severity of change. If the severity of change is small, the memory is used. Otherwise, a random restart is used instead of a memory. The algorithm uses this technique to guarantee a better convergence speed. The Multi-strategy Ensemble MOEA [60], similarly to [7], uses memory when a change occurs to obtain a new population by generating solutions randomly within the bounds of the search space and solutions generated by the Gaussian local search operator from the stored solution in that memory.

The second main class of techniques is the prediction-based algorithms. These algorithms aim to predict the new Pareto Front (i.e. the Pareto solutions) based on the information related to the prior history of changes. The prediction procedure is the main characteristic that differentiates a prediction-based algorithm from other algorithms. In the Population Prediction Strategy (PPS) proposed in [66], the authors divided the Pareto Solution (PS) into a center point and a manifold. When a change occurs, the algorithm predicts the next center from the history of all center points, and the next manifold is predicted from the previous manifolds. The manifold concept used in this paper is the same used in the RM-MEDA algorithm [65]. The authors of the Kalman Filter Assisted MOEA/D-DE Algorithm (MOEA/D-KF) [40] used a Kalman Filter-based prediction model, which is applied to the whole population to direct the search towards the new Pareto solution.

Authors in [64] proposed a Novel Prediction Strategies for Dynamic Multiobjective Optimization (PBDMO). The proposal combines the RM-MEDA algorithm with a new prediction-based reaction mechanism. PBDMO reacts effectively to any change occurs on the problem, by generating three sub-populations using different strategies.

Other methods have been proposed in the literature, notably the work entitled Reinforcement Learning-based Dynamic Multiobjective Evolutionary Algorithm (RL-DMOEA) proposed in [69], which uses Reinforcement learning technique to learn better evolutionary behaviors from environment information, when identifying different severity level in the problem changes.

2.3.2 Copula-based estimation of distribution algorithms

Copulas models have been used in Estimation of Distribution Algorithms (EDAs) [28]. An early application of copulas in EDAs was presented in [54], where the multivariate Gaussian copula was proposed to model dependencies between the variables. In that proposal, different types of univariate marginal distributions could be used for each variable. The introduced EDA was compared to the UMDA (Uni-variate Marginal Distribution Algorithm) on a set of

(static) single-objective functions showing superior results. Other variants of multi-variate Gaussian copula EDAs have been proposed in [23, 34].

A variant of Copulas, called Archimedean copulas, is also used in modeling EDAs. Initially, bi-variate Archimedean copulas were used to optimize problems in two dimensions [59]. Exchangeable Archimedean copulas are an extension of bi-variate Archimedean copulas dedicated to multi-variate distributions. The exchangeable Archimedean copulas were applied to problems of larger dimensions [22, 57]. Archimedean copulas have also been used to build EDAs that factorize the distributions [15].

The closest methods to our proposition are EDAs that use vine-copulas to model the distribution of the best solutions [45, 46, 52]. Specifically, authors in [52] have proposed C-vines (Canonical vines) and D-vines (Drawable vines) as variants of vine-copulas to model distributions in EDAs. Since the cost of constructing full vines increases with the number of variables, the authors proposed to use a truncation strategy, i.e., trees are only built-up to a given level. The strategy was based on metrics that penalize the complexity of the models. The D-vine EDA, proposed by Salinas et al. [45], was tested with several benchmark functions demonstrating good results. In [55], four undirected graphical models based on copula theory were investigated in the context of EDAs to address a molecular docking problem. The authors reported that the vine-based algorithms produce more efficient results than the other tested variants. Recently, some new proposals investigate the application of copula models to multiobjective problems [9, 10].

Though, Copulas and vine-copulas have been proposed for EDAs, None of these efforts was dedicated to use this in Dynamic Multiobjective Problems. Thus, we target DMOPs using Vine-copulas with the introduction of algorithms that uses a memory-like mechanism and a prediction-based scheme on the previous history of evolution.

To summarize, while previous applications of copulas and vine-copulas have been proposed, none of these models have been applied to dynamic multiobjective problems. Also, they do not use a memory-like mechanism or the previous history of the evolution as the algorithms introduced in this paper do.

3 A Copula-based estimation of distribution algorithm for dynamic MOPs

EDAs use many ways to estimate the distribution of the best solutions. Different methods of estimations are comprehensively introduced in [29]. In this context, copula-based models are good candidates to optimize complex functions [10, 44, 58]. We propose to use the copula model created in the estimation step to generate new solutions near the PS (Pareto Solutions) Thus, we introduce two approaches to find good solutions when a change occurs in the objective function. The first approach is a memory-based algorithm that uses the created copula model as a memory to store the best solutions found to the current generation before a change is detected. Then, use it as an initial population to guide the search. The second approach is a prediction-based algorithm that uses the history of the occurred changes to create a copula model in order to predict the next best solution.

3.1 The memory-based DynVC-EDA algorithm

The main idea of the memory-based algorithm is to use a memory of the best solutions obtained as an initial population instead of random initialization. This initialization is used when a change in the problem is detected. Mainly, memory-based algorithms reported in the literature [6] use an explicit memory. By explicit memory, we mean that the memory stores the best individuals found and then use them to create an initial population for the new configuration of the optimization problem. However, in our proposition, we use a vine-copula as a model of the best individuals found (i.e. The best solutions) as a memory. This concise representation reduces the size of the used memory. In addition, the model is expected to capture the common characteristic of the best solutions. Finally, generating new solutions using the model will add some random components that are useful for exploration. Therefore, this will help to diversify the initial population.

Algorithm 1: Dynamic Vine Copula-based EDA

```
1  Q_0 ← Initialization(N_0);
2  NDSet_0 ← Sorting(Q_0);
3  P_0 ← SelectFromNDS(N);
4  t ← 1;
5  while Not termination criteria do
6      if Change Detected then
7          C_t ← EstimateModel(P_{t-1});
8          P_tmp ← GenerateSolutions(C_t);
9      end
10     NDSet_t ← ApplyMOEAD(P_tmp ∪ NDSet_{t-1});
11     P_t ← SelectFromNDS(N);
12     NDSet_t ← P_t;
13     t ← t+1;
14 end
15 Return NDSet_t,C;
```

As a typical evolutionary algorithm, our proposed method has two main steps; the *Selection* and the *Reproduction*. In the first step, we use the MOEA/D to select the best solution, which will be used in the reproduction. In the second step, the copula model is applied to estimate then to generate new individuals. When a change occurs in the problem, the algorithm uses the obtained copula model to generate

individuals and use them as an initial population in the next generation. A pseudo-code of the estimation of distribution algorithm using a copula for a dynamic multiobjective problem is illustrated in Algorithm 1.

Initially, we create a population $\mathbf{P_0^t} = [\mathbf{x_1^t}, \ldots, \mathbf{x_m^t}]^T$ where $\mathbf{x_i^t}$, $i = 1, \ldots, m$ represents the i individual at time t. Every individual $\mathbf{x_i^t} = (x_{i1}^t, \ldots, x_{in}^t)$, where $x_{min} \leq x_{ij} \leq x_{max}$. In the selection step, we use the algorithms MOEA/D with Tchebycheff or Boundary Intersection (BI) [62]. This step returns a set of individuals that will be used in the reproduction. We call $\mathbf{P_{sel}^t}$ the set of the individuals at time t resulting from the selection process operated on the precedent population. For the first generation, we use the initial population at time $t = 0$, $\mathbf{P^0}$. We have $\mathbf{P}$ defined as the following:

$$\mathbf{P_{sel}^t} = \begin{matrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{matrix} \tag{22}$$

The reproduction step contains two main stages, the creation of the copula model and the generation of the individuals. First, we create the copula model by considering every individual $\mathbf{x_i^t}$ as an observation i. Thus, we see the set $\mathbf{P_{sel}^t}$ as a set of observations of a vector of random variables. This set of observations is considered to follow a joint distribution function. The copula model creation step aims to estimate this joint distribution.

As we have previously explained, the inputs of the first step are the n-dimensional selected solutions, so we should use a multi-variate copula to model it. However, there is a different way to model a multi-variate copula [35]. In this work, we use a class of copula known as Vine-Copula, especially C-vine class, and the vine is truncated in the first level to gain more efficiency.

We start the creation of the model step by estimating the parameters for all bi-variate copulas, as it is described in [35]. There exist many methods to estimate copula parameters. Those methods include the fully parametric maximum likelihood method [61], the semi-parametric maximum pseudo-likelihood (MPL) method [25], the inference functions for margins method [35], the inversion of Kendall's tau estimator [61], and the inversion of Spearman's ρ estimator [61]. We select to use in this paper the MPL method since it has outperformed other methods in several studies [36, 56].

After estimating the parameters of the vine-copula, we proceed to the sampling step. First, we use the modeled vine-copula to generate samples $U_i = (u_{1i}, u_{2i}, \ldots, u_{in})$, where $u_{ij} \in [0, 1]$ as described in Sect. 2.2.4 by the same procedure as described in [35]. Then, we take the generated U_i values and fit them to the solutions space Ω.

To fit the generated values to the space solutions Ω, we use a Kernel Density Estimation method (KDE) [49]. The kernel density estimation takes a set of observations and then estimates for every observation a probability value according to a specific kernel function [49]. To do this, we create the set O as follows:

$$\begin{aligned} X_{sel} &= \{X^1, X^2, \cdots, X^K\} \\ X^j &= (x_0^j, x_1^j, \cdots, x_n^j), \text{ where } x_i^j \in [x_{i_{min}}, x_{i_{max}}] \\ O &= \{o_1, o_2, \ldots, o_n\}, \text{ where } o_l = \{x_{l1}, x_{l2}, \ldots, x_{lK}\} \end{aligned}$$

We denote as o_l observations which contain the elements of the column l of the solution in the set X_{sel} (selected solutions). We estimate the density of the observation in o_l using the Kernel Density Estimation (KDE) method. The result of the KDE for the o_l observations is a set $\hat{V}_l = \{\hat{v}_{l1}, \hat{v}_{l2}, \ldots, \hat{v}_{lK}\}$ where $\hat{v}_{li}$ represents the estimated density of the observation o_{li}. After estimating the density of all the set O using the same procedure that we used to estimate the o_l observations, we start the step of interpolation.

The aim of the interpolation step is to create an inverse cumulative distribution function (icdf). The icdf function is defined from [0, 1] to $\mathbb{R}$. For the interpolation method, we take the set $\hat{V}_l$ as an input X vector and o_l as an output Y vector. We get a function f_l that gives $x_{li} = f_l(\hat{v}_{li})$ where $\hat{v}_{li} \in [0, 1]$ and $x_{li} \in [x_{l_{min}}, x_{l_{max}}]$.

After creating the interpolation function, labeled *inv_cdf* in Algorithm 2, we produce the new solutions using the generated sample U and the *inv_cdf* function. For every column l, we apply a function f_l for all elements in U_l, where U_l are the elements of the l-th parameters for all the set U. The application of f_l gives us a set X'_l where $X'_l = \{x'_{l1}, x'_{l2}, \ldots, x'_{lK'}\}$ and K' is the number of the sampled solutions, $x'_{li} = f_l(U_{li})$. The result of this step is n number of vector, and the new population is:

$$P' = \left[X'_1, X'_2, \ldots, X'_n\right] \tag{23}$$

$$P' = \begin{bmatrix} x'_{11} & \cdots & x'_{1n} \\ \vdots & \ddots & \vdots \\ x'_{K'1} & \cdots & x'_{K'n} \end{bmatrix} \tag{24}$$

a row $(x'_{i1}, x'_{i2}, \ldots, x'_{in})$ from the matrix P' represents a newly generated solution.

We use the result of the density estimation to create an interpolation function between the values in [0, 1] and its probability density function (inverse CDF). Then, we apply this function for every generated value obtained by sampling the vine-copula. The result of this step is a new set of generated solutions in Ω. Algorithm 2 summarizes the essential steps of this phase.

Algorithm 2: Generate Solutions

```
for i in {0, 1, ··· , n} do
    y = kernel_density(X_i);
    inv_cdf = interpolate(y, X_i);
    x = inv_cdf(ux_i);
    Result.append(x);
end
return Result;
```

3.2 The prediction-based DynVC-EDA algorithm

In this section, we describe how the algorithm reacts when the problem is changed. Our algorithm uses a prediction-based scheme to reinitialize a population to guide the search process. The basic idea of the prediction-based algorithm is to utilize history information to predict an initial population that is close to the new PS. To achieve this goal, we store - at least two changes - the history of the best solutions found up to the current generation. And when a new change is detected, we use the saved set of solutions to create a copula model. This copula model is employed to predict the new Pareto solution and use it as an initial population.

Similarly to the memory-based algorithm, we use vine-copula due to its ability to model a high dimensional dependency. The type of copula used is a C-vine copula truncated at the first level. As it is described in Sect. 2.2.4, every node in the tree of the vine is a bi-variate copula. Every bi-variate copula C_i models the dependency between the ith parameter of all PS at time $t-1$ and the ith parameter of all PS at time t.

Let PS^{t-1} and PS^t be the PS at the time $t-1$ and t respectively. We mean by t the time when a change occurs. The method consists of creating n bi-variate copulas

$$Model(PS^{t-1}, PS^t) = (C_0, C_1, \ldots, C_n).$$

For example, C_0 model the dependence between x_{j0}^{t-1} and x_{j0}^t where $j \in \{0, 1, \ldots, k\}$, k is the number of the selected solutions in the PS. If we take the same notation in Eq. (23) that represents the population as a matrix, for every copula $C_i(X_i, Y_i)$, X_i that represents the column i of the matrix representation of PS at time $t-1$ and Y_i that represents the column i of the matrix representation of PS at time t. The result of this modeling process is n copulas, and the next step is the use of the set of copulas to generate the new solutions.

4 Experimentation results and analysis

In this section, we report the experimentation results conducted for evaluating the diversity and convergence of the proposed algorithms (DynVC-EDA-M and DynVC-EDA-P).

4.1 Benchmark function for dynamic optimization problem

Some of the significant challenges in the field of dynamic multiobjective optimization (DMOO) are the lack of a standard set of benchmark functions. Recently, some of these challenges have been addressed: a comprehensive overview of existing dynamic multiobjective optimization problems was presented in [31]. The characteristics of an ideal benchmark function suite were proposed with a set of DMOPs are suggested in [31]. Moreover, a comprehensive overview of performance measures currently used for DMOO is detailed in [32, 33]. In addition, authors in [24] developed a number of GTA problems targeting dynamic modality, trade-off connectivity, and PF degeneration.

To test the efficiency of the proposed algorithms, we conducted different experiments using a set of benchmarks

Table 1 The characteristics of the used benchmarks in the experiments

Benchmark	Characteristics
GTA1	Convex, uni-modal(landscape modality)
GTA2	Convex, multi-modal
GTA3	Convex, mixed uni and multi-modal
GTA4	Disconnected, mixed uni and multi-modal
GTA5	Mixed convex and disconnected, multi-modal
GTA6	Mixed convex and disconnected, mixed uni and multi-modal
FDA4	Non-convex
FDA5	Non-convex, spread of solutions changes over time
DIMP2	Convex, each decision space has its own rate of change
DMOP2	Changes from convex to concave and vice versa
DMOP3	Convex POF, spread of the POF solutions changes over time
HE2	Discontinuous POF, with various disconnected continuous sub-regions
HE7	POF changes from convex to concave, and vice versa
HE9	POF changes from convex to concave and vice versa

Table 2 The mean value of the MIGD for the 30 runs on a set of the CEC2015 benchmarks

n_t	τ_t	Max gen (τ_T)	Benchmarks	DynVC-EDA-P	DNSGA2	MOEAD	RND
10	5	100	DIMP2	**0.37288180**	0.39959050	0.37878240	0.39357860
			DMOP2	**0.02875668**	0.06711308	0.12026150	0.11155490
			DMOP3	0.00882665	**0.00839224**	0.01171157	0.01253386
			FDA4	**0.00275787**	0.00443703	0.00420980	0.00863301
			FDA5	**0.00673678**	0.00757499	0.01225450	0.02009404
			HE2	0.00206986	**0.00198381**	0.00199345	0.00208311
			HE7	0.02322992	**0.00248262**	0.00422058	0.00931802
			HE9	0.02339533	**0.00665999**	0.00839795	0.01201706
10	10	200	DIMP2	**0.36455740**	0.43013310	0.39320480	0.38912460
			DMOP2	**0.02261942**	0.05323162	0.04597937	0.11021240
			DMOP3	0.00905562	**0.00806832**	0.01057286	0.01257349
			FDA4	**0.00243462**	0.00383710	0.00290031	0.00856885
			FDA5	**0.00531760**	0.00687733	0.00632778	0.02349795
			HE2	0.00209184	**0.00198256**	0.00199128	0.00209045
			HE7	0.02522046	**0.00240826**	0.00387559	0.00919402
			HE9	0.02386652	**0.00669293**	0.00809058	0.01188517
10	25	500	DIMP2	**0.39198060**	0.46529780	0.49638490	0.39260760
			DMOP2	**0.01951606**	0.03125757	0.02450468	0.10874850
			DMOP3	0.01064414	**0.00721796**	0.01183724	0.01260677
			FDA4	**0.00207008**	0.00331563	0.00214757	0.00867243
			FDA5	0.00462470	0.00453550	**0.00443193**	0.02023647
			HE2	0.00209877	**0.00198219**	0.00199044	0.00208618
			HE7	0.03814778	**0.00229699**	0.00339025	0.00925746
			HE9	0.02412944	**0.00663140**	0.00783940	0.01191000

Bold indicates the best value

Table 3 Mean of the MIGD metric for the dynamic CEC2015 benchmarks for $n_t = 10$, $\tau_t = 25$ and $\tau_T = 500$ in 30 runs

Benchmarks	DynVC-EDA-M	DynVC-EDA-P	DNSGA2	MOEAD	RMMEDA	RND
DIMP2	0.40075930	0.39198060	0.46529780	0.49638490	**0.20212650**	0.39260760
DMOP2	0.03277817	0.01951606	0.03125757	0.02450468	**0.00185622**	0.10874850
DMOP3	0.01542056	0.01064414	**0.00721796**	0.01183724	0.02112254	0.01260677
HE2	0.00205796	0.00209877	**0.00198219**	0.00199044	0.00212932	0.00208618
HE7	0.01044204	0.03814778	**0.00229699**	0.00339025	0.00105565	0.00925746
HE9	0.01659027	0.02412944	0.00663140	0.00783940	**0.00563392**	0.01191000

Bold indicates the best value

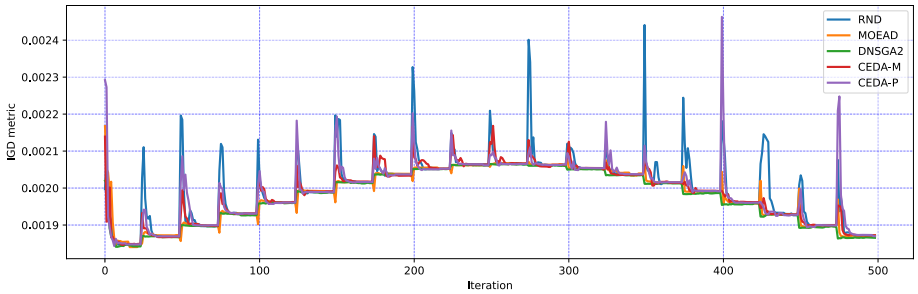
that are usually used to evaluate new solving algorithms in the area of dynamic multiobjective optimization. The CEC2015 set of benchmarks introduced in [30] provides a set of benchmarks to compare the new algorithms with the classical ones. More specifically, we have selected the FDA4, FD5, DIMP2, DMOP2, DMOP3, HE2, HE7 and HE9 benchmarks from CEC2015. In addition to the previous benchmarks, we tested the proposal on a new set of benchmark named GTA benchmarks presented in [24]. For all experiments, we perform 30 runs with different combinations of parameters.

4.2 Performance measures

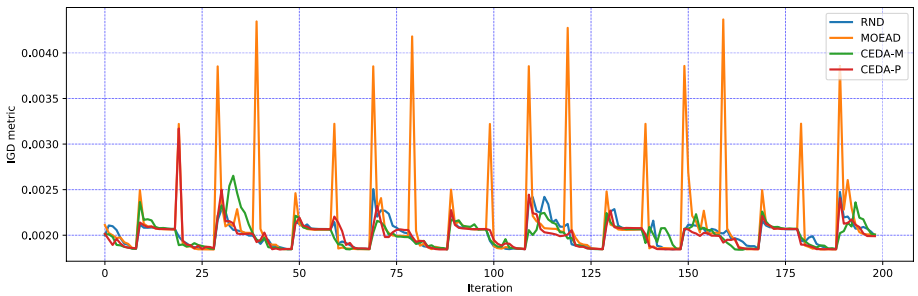
To assess the efficiency of the proposed algorithm, we apply an extensively used quality indicator: the inverted generational distance (IGD) for all used benchmark.

Let P^{t*} be a set of uniformly distributed Pareto optimal points in the PF^t. Let P^t be an approximation of PF^t. The IGD metric is defined as:

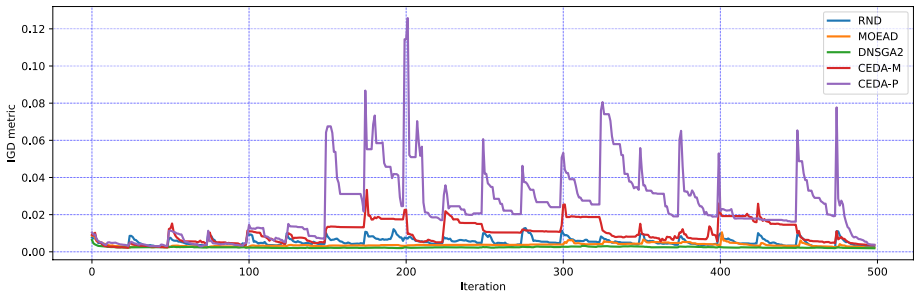
$$IGD(P^{t*}, P^t) = \frac{\sum v \in P^{t*} d(v, P^t)}{|P^{t*}|} \tag{25}$$



(a) HE2, $n_t = 10$, $\tau_t = 25$



(b) HE2, $n_t = 1$, $\tau_t = 10$



(c) HE7, $n_t = 10$, $\tau_t = 25$

Fig. 2 IGD over iteration for the HE2 and HE7 benchmarks with different n_t and τ values

where

$$d(v, P^t) = \min_{v \in P^{t*}} ||F(v) - F(u)|| \tag{26}$$

is the distance between v and P^t, and $|P^{t*}|$ is the cardinality of P^{t*}. The IGD metric can measure both diversity and convergence. To have a low IGD value, P^t must be very close to PF^t and cannot miss any part of the whole PF^t [66].

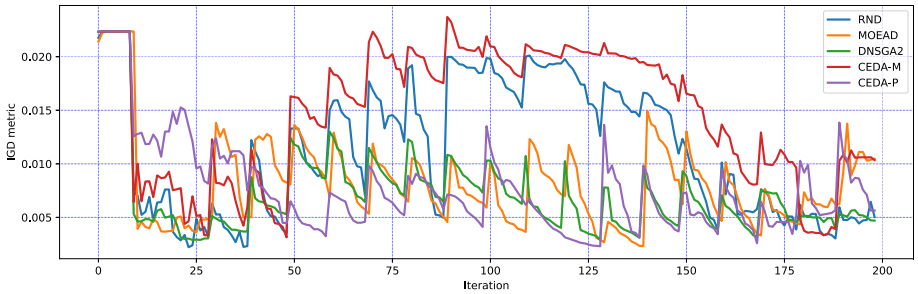
We use the average of the IGD values "MIGD" (Mean IGD) in a number of steps over a run as described in [66]. The MIGD metric is given by equation:

$$MIGD = \frac{1}{|T|} \sum_{t \in T} IGD(P^{t*}, P^t) \tag{27}$$

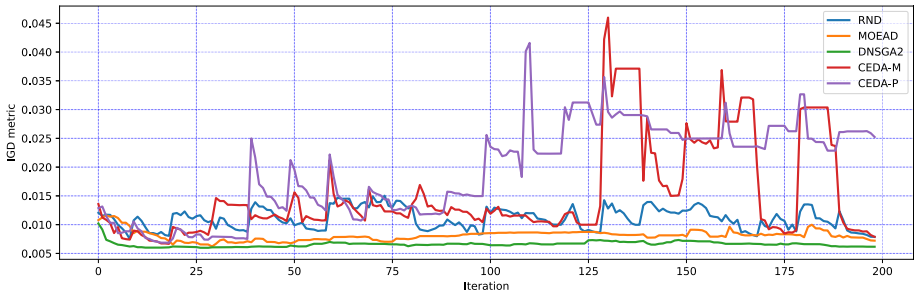
where T is a set of discrete-time points in a run, and $|T|$ is the cardinality of T. A lower value of the modified inverted generational distance (MIGD) metric would assist in evaluating the tracking ability, as the approximated Pareto front obtained from the algorithm with the changing POF is measured before every change.

4.3 Parameter settings and algorithms

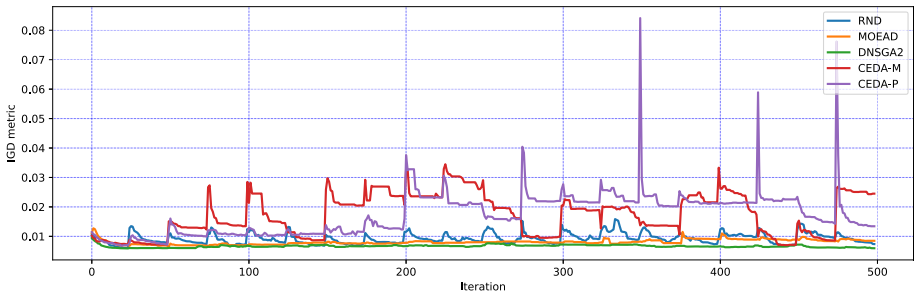
The parameters of the used MOEAs in the experiment were referenced from their original papers. Some key parameters in these algorithms are set as follows:



(a) DMOP3, $n_t = 10$, $\tau_t = 10$



(b) HE9, $n_t = 10$, $\tau_t = 10$



(c) HE9, $n_t = 10$, $\tau_t = 25$

Fig. 3 IGD over iteration for the HE9 and DMOP3 benchmarks with different n_t and τ values

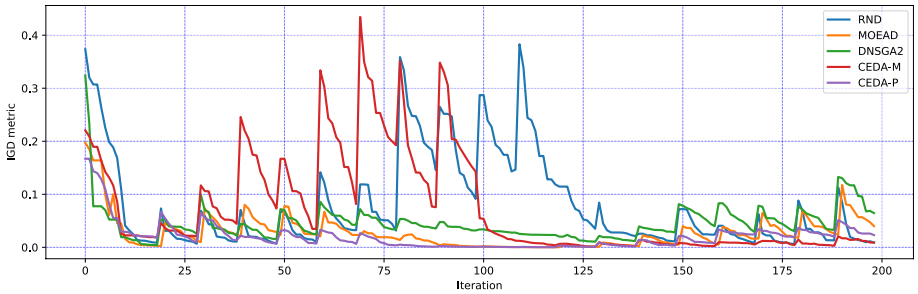
- *Population size* The number of individuals we use in the experimentation is set to 100 for all algorithms.
- *Maximum number of generations* We denote this parameter as τ_T, and we consider $\tau_T \in \{100, 200, 500, 1000\}$
- *Dynamic configuration* (n_t, τ_t) For these parameters, we use the same configuration used in CEC2015 competition [30].

In addition to the proposed DynVC-EDA-M and DynVC-EDA-P algorithms, we use other algorithms, from competitive methods, to show the quality of the proposal and compare results with state-of-the-art algorithms. These competitive algorithms are the following:

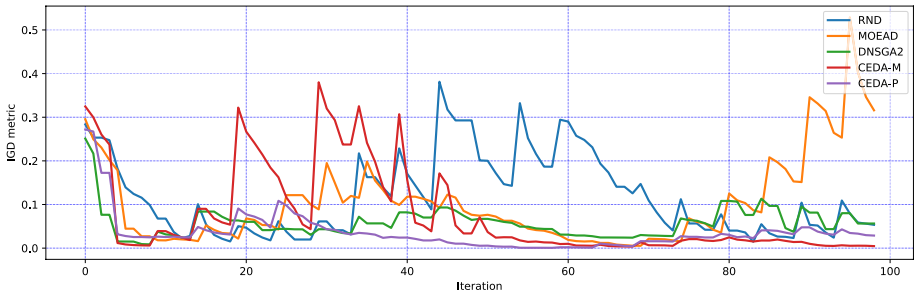
- *RM-MEDA* It is used in the PPS algorithm [66].
- *RND* In this algorithm, the population is randomly initialized every time a change occurs.
- *DNSGA2* A dynamic version of the NSGA2 proposed by Deb et al. in [17].
- *MOEA/D* The variant used in [67]. It continues the execution without considering the changes.

5 Results and discussions

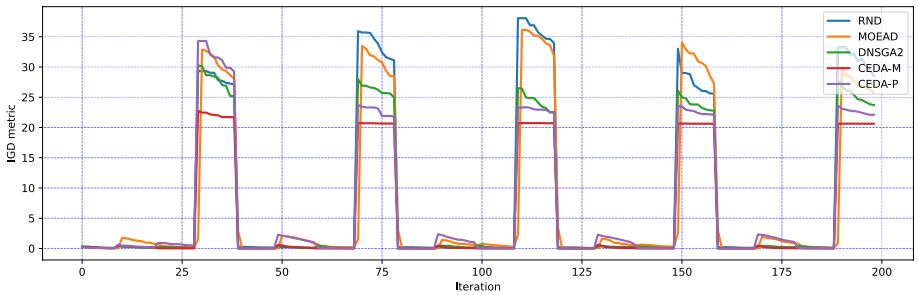
We show in this section the results obtained from the experiments. The first set of experiments is conducted using the CEC2015 benchmarks, and the second one uses the GTA(a,m) benchmarks. Table 1 presents the characteristics of all used benchmarks in many points of view, such as the



(a) $n_t = 10,\ \tau_t = 10$



(b) $n_t = 10,\ \tau_t = 5$



(c) $n_t = 1,\ \tau_t = 10$

Fig. 4 IGD over iteration for the DMOP2 benchmark with different n_t and τ values

convexity, modality, POS changes, and the continuity of the Pareto Optimal Front (POF).

All the benchmarks are related to a time variable t. The time variable (t) used in all functions is calculated using the current generation number denoted as (τ) by

$$t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \tag{28}$$

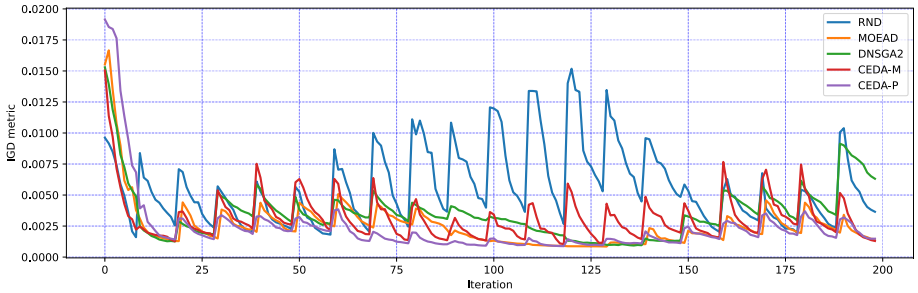
where n_t is the number of distinct steps in one t, also known as the severity of change. τ_t represents the number of generations for which the fitness landscape remains fixed, and it is called the change frequency.

5.1 Results on CEC2015 benchmarks

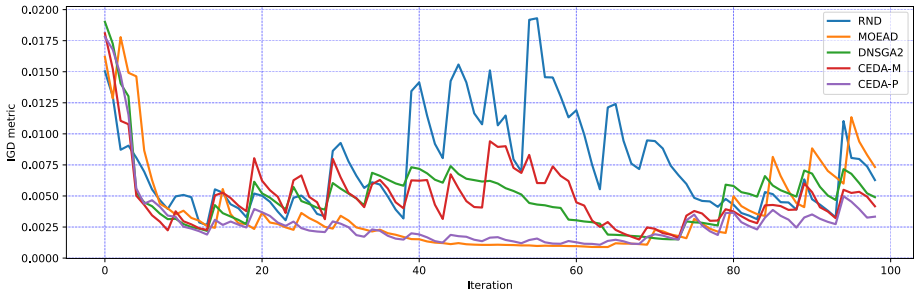
The CEC2015 set of benchmarks contains many types of problems. To investigate the behavior of the proposal on those benchmarks, we perform 30 runs and compare the obtained results for all the algorithms described in Sect. 4.3.

For the configuration $n_t = 10, \tau_t = 5$ and $\tau_T = 100$ DynVC-EDA-P outperforms all the other algorithms on DIMP2, DMOP2, FDA4 and FDA5 benchmarks and get a good result for the DMOP3 and HE2 benchmarks with approximately the same best value obtained by the DNSGA2 algorithm. We can notice for the other configurations in Table 2, that the value of the MIGD is decreased according to the value of τ_t. As an example, for the FDA4 the MIGD values are 0.0027, 0.0024 and 0.0020 for τ_t equal to 5, 10 and 25 and the same conclusion still correct for the rest of the benchmarks.

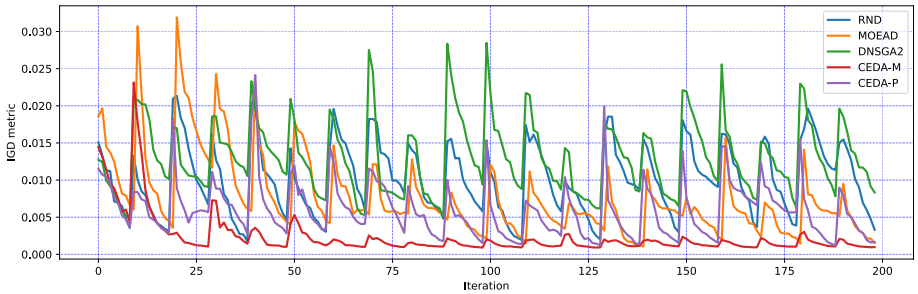
Table 3 presents the obtained results using $n_t = 10$ and $\tau_t = 10$. The result shows that the two variants of the



(a) $n_t = 10$, $\tau_t = 10$



(b) $n_t = 10$, $\tau_t = 5$

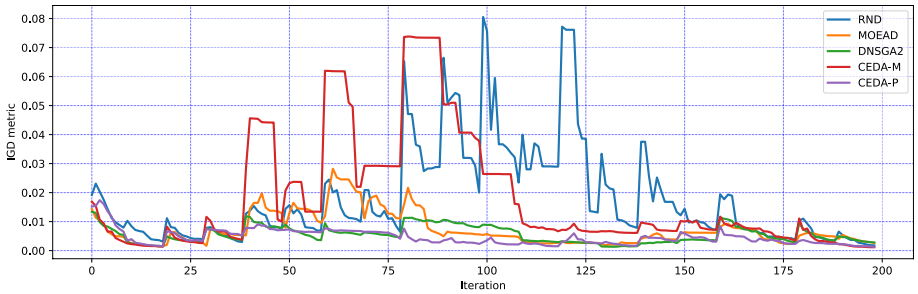


(c) $n_t = 1$, $\tau_t = 10$

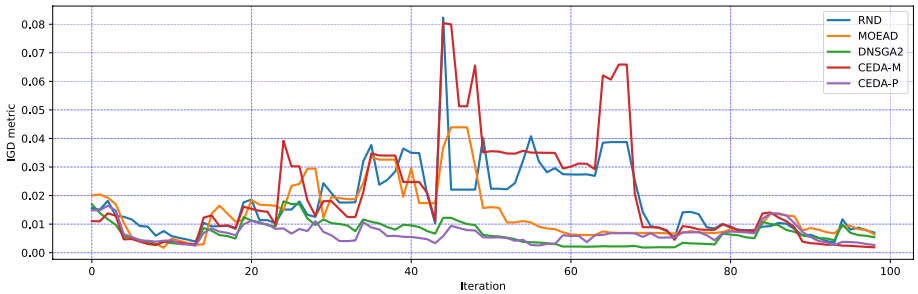
Fig. 5 IGD over iteration for the FDA4 benchmark with different n_t and τ values

proposed algorithm are competitive compared to state-of-the-art algorithms. The RM-MEDA algorithm outperforms the proposed algorithms by a small value. For example, in the HE2 benchmark the obtained MIGD is 0.0020 for both DynVC-EDA-P and DynVC-EDA-M, and the MIGD value for the RM-MEDA is 0.0019 which is approximately the same result. However, the two algorithms DynVC-EDA-P and DynVC-EDA-M outperform RM-MEDA in the remaining benchmarks.

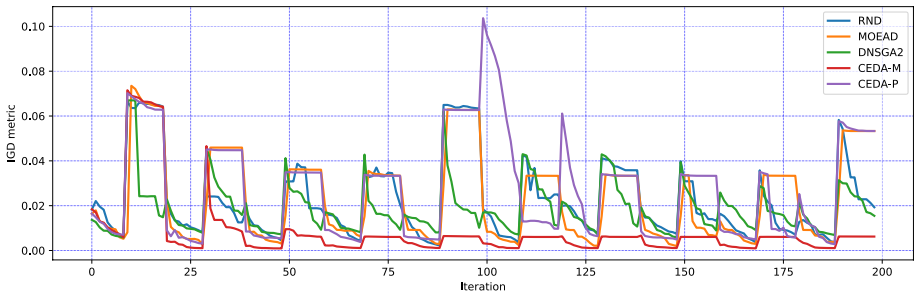
In addition to the mean of MIGD results depicted in Tables 2 and 3, the variation of the IGD metric over iterations are plotted from Figs. 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 to 13. Each figure represents the behavior of the compared algorithms against a specific benchmark when changes occur in the problem. It is important to highlight that as long as the IGD value is decreasing or stable after reaching a low value then the algorithm in question is performing good. In other terms, we can qualify that an algorithm achieves a good PS approximation for DMOPs, if its IGD value remains stable or increases slightly when a change occurs over a period of time. In contrast, the



(a) $n_t = 10,\ \tau_t = 10$



(b) $n_t = 10,\ \tau_t = 5$



(c) $n_t = 1,\ \tau_t = 10$

Fig. 6 IGD over iteration for the FDA5 benchmark with different n_t and τ values

algorithm is not appropriate for DMOPS if its IGD value significantly increases when a change occurs.

Figure 2 presents the variation of the IGD metric over iterations (generation). Figure 2a and b depict results for the HE2 benchmark, and Fig. 2c shows the IGD values for the HE7 benchmark.

The variation of IGD value in Fig. 2a and b shows that DynVC-EDA-M and DynVC-EDA-P (labeled in the plot as CEDA-M and CEDA-P respectively) can adapt rapidly when a change occurs for the different configuration in HE2 benchmark. For instance, DynVC-EDA-P reached the best IGD value, among other methods, after the 25 iterations (when the change occurred) for the HE2 benchmark as presented in Fig. 2b.

Table 4 The mean value of the MIGD for the 30 runs on the GTAm benchmarks

n_t	τ_t	Max Gen (τ_T)	Benchmarks	CEDA-M	CEDA-P	DNSGA2	MOEAD	RND
10	5	100	GTA1m	0.00504107	**0.00337551**	0.01298300	0.00864847	0.04299610
			GTA2m	0.00646488	**0.00456420**	0.02089659	0.01305241	0.06021659
			GTA3m	0.02192453	**0.00720261**	0.04611431	0.02372928	0.08898506
			GTA4m	0.05724218	**0.03309505**	0.09469108	0.09139035	0.21478290
			GTA5m	0.01690705	**0.00622164**	0.02702857	0.02069996	0.08526735
			GTA6m	0.03246029	**0.01263020**	0.06547793	0.03361206	0.13165150
10	10	200	GTA1m	0.00544174	**0.00308445**	0.00868691	0.00389169	0.04326112
			GTA2m	0.00639586	**0.00398748**	0.01309809	0.00612454	0.06071674
			GTA3m	0.02088232	**0.00638953**	0.03425049	0.01194394	0.08852944
			GTA4m	0.05685193	**0.02339158**	0.06543739	0.06263873	0.21566660
			GTA5m	0.01855513	**0.00564686**	0.01675821	0.00896819	0.08607723
			GTA6m	0.03263400	**0.01071536**	0.05059991	0.01991099	0.13182840
10	25	500	GTA1m	0.00597791	**0.00254588**	0.00447338	0.00233712	0.04348401
			GTA2m	0.00750844	**0.00327940**	0.00638779	0.00353216	0.05996574
			GTA3m	0.01635174	**0.00535516**	0.01535604	0.00903214	0.08942659
			GTA4m	0.05032064	**0.01966904**	0.03504405	0.03401721	0.21397000
			GTA5m	0.02413267	**0.00506411**	0.00841566	0.00506903	0.08759173
			GTA6m	0.03299843	**0.00910212**	0.02454214	0.01463475	0.13231680
10	50	1000	GTA1m	0.00611523	**0.00228844**	0.00302969	0.00182793	0.04332168
			GTA2m	0.00854330	**0.00293369**	0.00446103	0.00296120	0.06049085
			GTA3m	0.01506768	**0.00511731**	0.01068644	0.00837761	0.08943080
			GTA4m	0.04438731	**0.01812110**	0.02966743	0.02448345	0.21394460
			GTA5m	0.02293416	**0.00461490**	0.00614030	0.00422840	0.08509316
			GTA6m	0.03353136	**0.00884431**	0.01928647	0.01321493	0.13274190

Bold indicates the best value

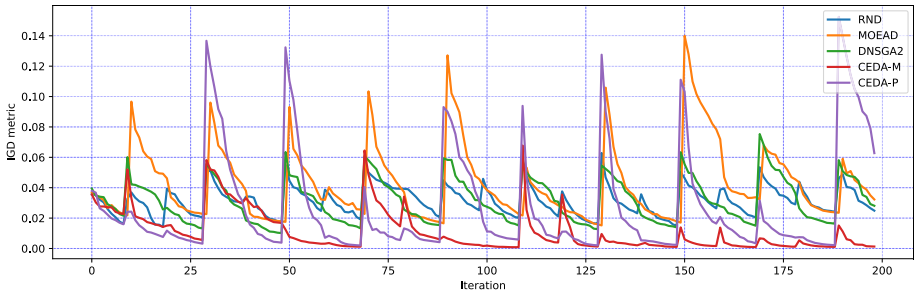
Table 5 The mean value of the MIGD for the 30 runs on the GTAa benchmarks

n_t	τ_t	Max Gen (τ_T)	Benchmarks	CEDA-P	DNSGA2	MOEAD	RMMEDA	RND
10	5	100	GTA1a	**0.00784733**	0.03866187	0.02450829	0.01235395	0.07586533
			GTA2a	**0.01094229**	0.05289864	0.04859465	0.01749281	0.10554060
			GTA3a	0.01752161	0.11179470	0.06421525	**0.01215462**	0.18611760
			GTA4a	0.08030599	0.25751840	0.24429410	**0.04154539**	0.47355140
			GTA5a	**0.01313602**	0.07906827	0.07515695	0.02090624	0.14881050
			GTA6a	0.03094385	0.16091150	0.07750096	**0.01366613**	0.27130630
10	10	200	GTA1a	**0.00562984**	0.03544574	0.00940386	0.00725997	0.07691511
			GTA2a	**0.00762695**	0.04607330	0.01430037	0.00993712	0.10609520
			GTA3a	0.01405916	0.10028970	0.02308525	**0.00671177**	0.18522820
			GTA4a	0.06744207	0.21757310	0.11866630	**0.02844747**	0.46515060
			GTA5a	**0.01012365**	0.06799495	0.01792568	0.01190752	0.14650190
			GTA6a	0.02404051	0.14433650	0.03714723	**0.00830946**	0.27171430

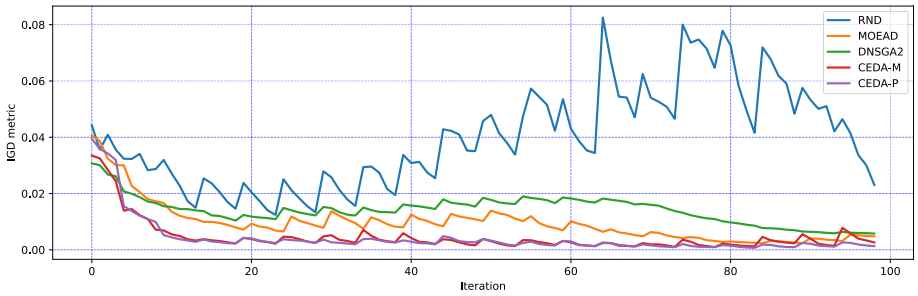
Bold indicates the best value

Similarly to Figs. 2 and 3 shows how close is the approximated Pareto front, obtained by each algorithm, to the Pareto Optimal Front (POF) in every iteration.

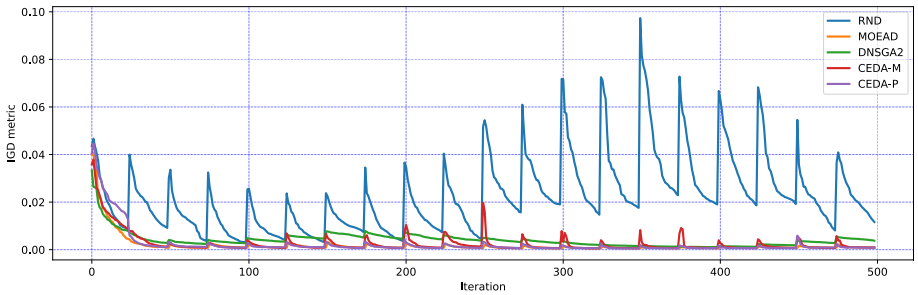
Figures 2c and 3b, c show that DynVC-EDA-M and DynVC-EDA-M get a good IGD value (the best PF approximation among others), and when a change occurs DynVC-EDA-M and DynVC-EDA-M maintain approximately the same IGD value, when the other algorithms show an



(a) $n_t = 1$, $\tau_t = 10$



(b) $n_t = 10$, $\tau_t = 5$



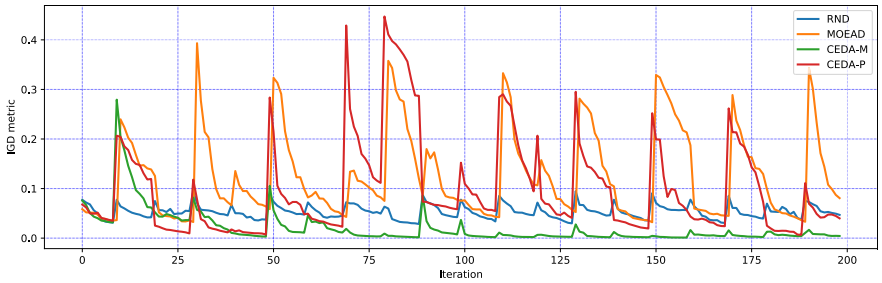
(c) $n_t = 10$, $\tau_t = 25$

Fig. 7 IGD over iteration for the GTA1m benchmark with different n_t and τ values

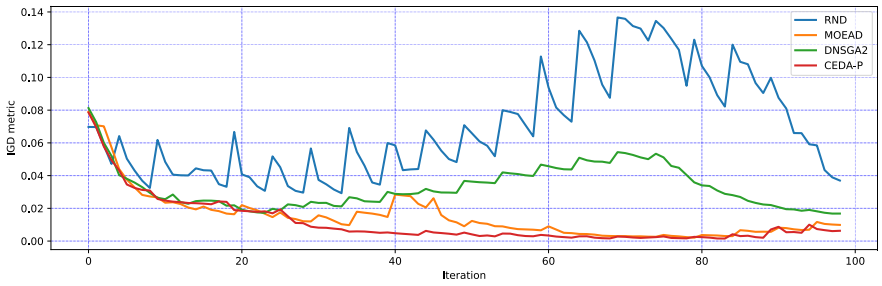
important increase of the IGD metric. Moreover, in both Figs. 2a and 3a, all the compared algorithms did not obtain a good approximation when the change of the problem occurs. This is due to the nature of DMOP3 problem where the spread of the POF solutions changes over time as shown in Fig. 2a. Thus, all the algorithms cannot adapt effectively to changes due to the severity of change occurrence in the problem within this experiment.

Figure 4 presents the variation of IGD for the DMOP2 benchmark for different configurations. In Fig. 4a, we present the result obtained by using $\tau_t = 10$. This figure shows that DynVC-EDA-P (label as CEDA-P in the figure) outperforms all the compared algorithms. The DynVC-EDA-P algorithm adapts in an effective way to the change of the problem and gets better results starting from iteration 75. The proposed DynVC-EDA-M algorithm (label as CEDA-M in the figure) gets better results starting from iteration 100, where the algorithm has more information about solutions on its memory, which can be effectively employed to adapt to the problem dynamic.

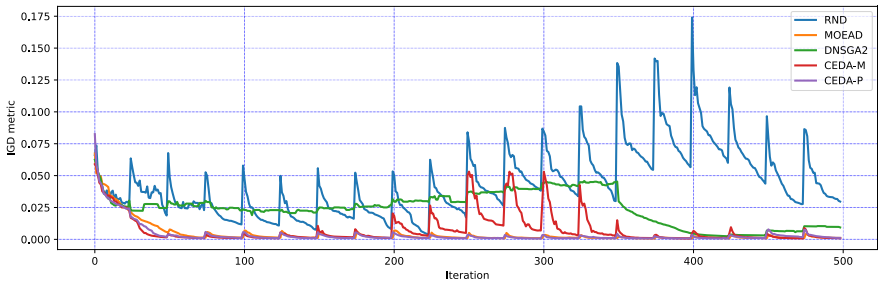
The same observation and conclusion remain correct for results in Fig. 4b which represents the IGD value for



(a) $n_t = 1,\ \tau_t = 10$



(b) $n_t = 10,\ \tau_t = 5$



(c) $n_t = 10,\ \tau_t = 25$

Fig. 8 IGD over iteration for the GTA1a benchmark with different n_t and τ values

DMOP2 benchmark using $\tau_t = 5$. As we can see, the proposed algorithms can adapt to problem change after 40 and 50 iterations for the memory-based and the prediction-based algorithm, respectively.

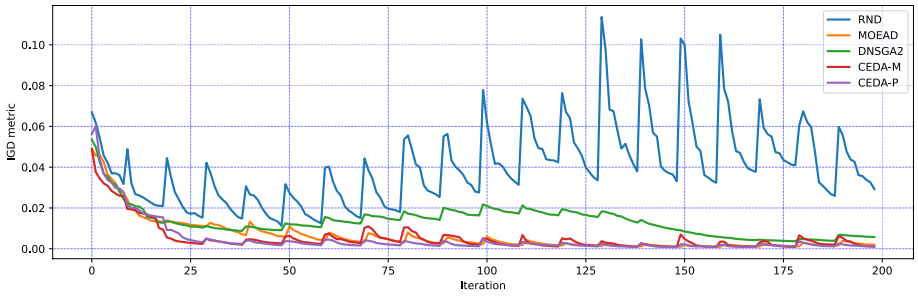
Figure 5 depicts the result of the experiment using the FDA4 benchmark. As it is shown in Fig. 5c, DynVC-EDA-M reaches a steady IGD value over time compared to other methods regardless the number of changes occurred on the problem. The same conclusion can be driven from Fig. 6 for the FDA5 benchmark. Using $\tau_t = 5, 10$, and $n_t = 10$ with FDA4 and FDA5 benchmarks and for both DynVC-EDA-M and DynVC-EDA-P, the IGD value remains stable when change occurs on the problem. However, other algorithms like RND or MOEA/D did not reach good stability. In addition, both DynVC-EDA-M and DynVC-EDA-P outperforms DNSGA2 as depicted in the two figures.

5.2 Results on GTA benchmarks

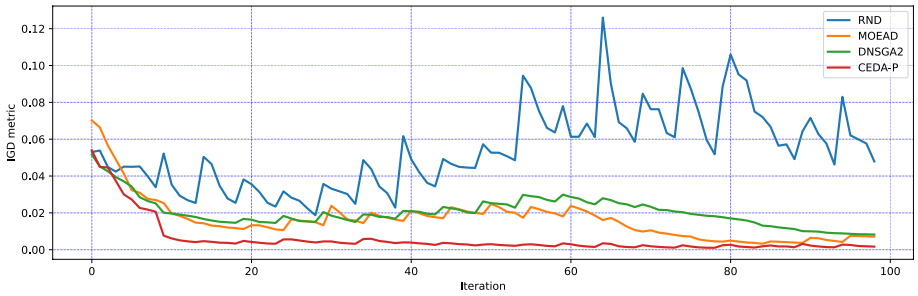
The GTAa (addition) and GTAm (multiplication) benchmarks are a class of GTA benchmarks proposed in [24] (the source code of the GTA benchmark and the Pareto front generator can be obtained from the author website.[1]) The set of GTA benchmarks ensures many characteristics such as convexity, modality, and discontinuity of the Pareto Front, as shown in Table 1.

For all experiment instances on the GTA benchmarks, we performed 30 runs for each parameter configuration. We calculated the mean of the MIGD values, and we plotted the change of the IGD value over iterations for each benchmark. Table 4 presents these results. This table shows that the proposed DynVC-EDA-P achieves the lowest MIGD

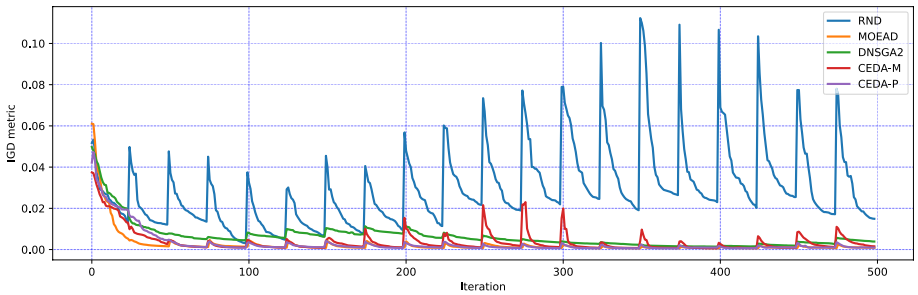
[1] https://github.com/senbong87/dmo_benchmark.



(a) $n_t = 10,\ \tau_t = 10$



(b) $n_t = 10,\ \tau_t = 5$



(c) $n_t = 10,\ \tau_t = 25$

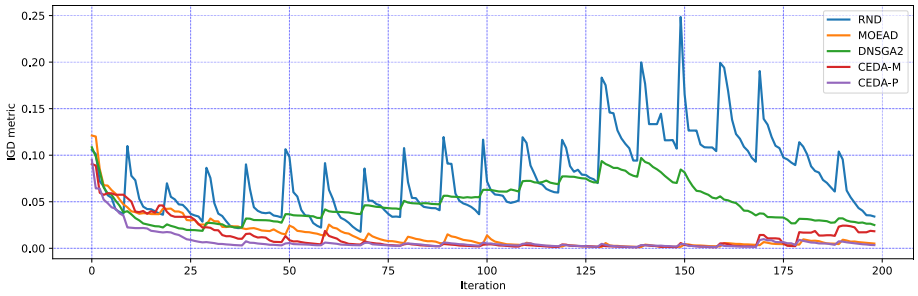
Fig. 9 IGD over iteration for the GTA2m benchmark with different n_t and τ values

value against the other algorithms for the GTA1m, GTA2m, GTA3m, GTA4m, GTA5m, and GTA6m benchmarks.

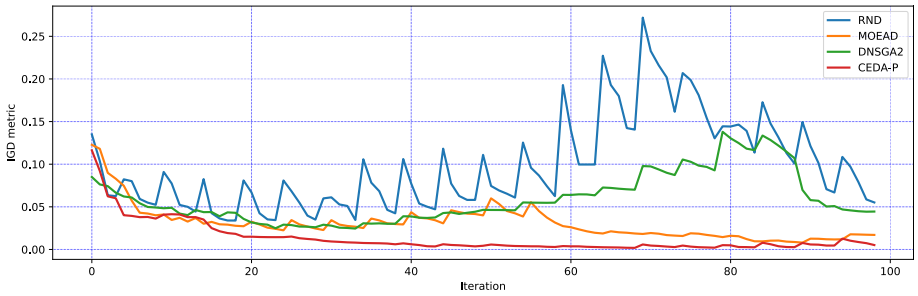
Moreover, for the GTAa addition results resumed in Table 5, the DynVC-EDA-P algorithm outperforms the other algorithms by achieving the lowest MIGD for the GTA1a, GTA2a and GTA5a benchmarks using the settings $n_t = 10$, $\tau_t = 5, 10$. Though the RM-MEDA algorithm obtains good MIGD for GTA3a, GTA4a and GTA6a benchmarks when the problem is mixed modality (uni and multi-modal), the DynVC-EDA-P algorithm obtains good results when the problem is convex and multi-modal or uni-modal.

In addition to the results presented in Table 5, we also plotted the variation of the IGD over iterations for each GTA benchmark. Plotting the IGD over iteration aims to investigate how the algorithms behave when a change occurs during the process of optimization.

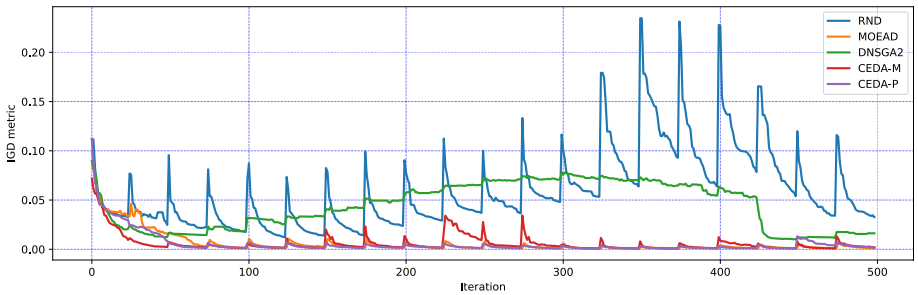
Figure 8 shows the obtained results for the GTA1a (addition) benchmark. Figure 8b presents the results when $\tau_t = 5$ and it shows that the proposed DynVC-EDA-P (label as



(a) $n_t = 10$, $\tau_t = 10$



(b) $n_t = 10$, $\tau_t = 5$



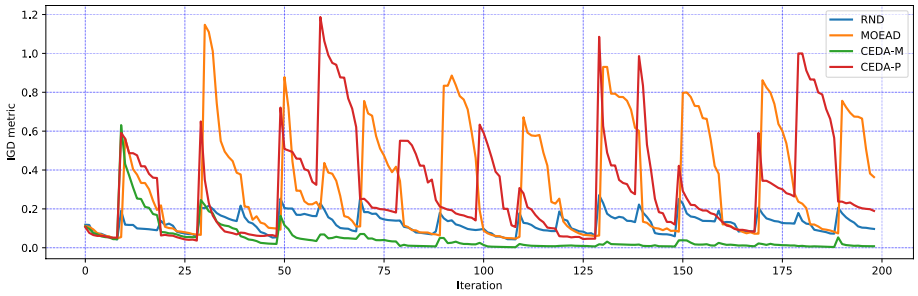
(c) $n_t = 10$, $\tau_t = 25$

Fig. 10 IGD over iteration for the GTA2a benchmark with different n_t and τ values

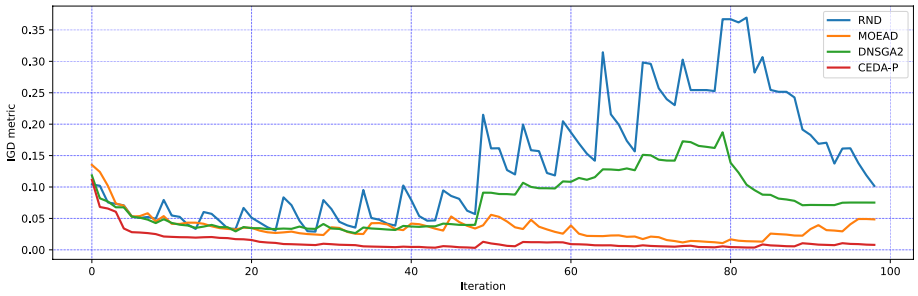
CEDA-P in the plot) can adapt effectively to the problem change and thus gets a better IGD value against the other algorithms starting from the 40-th iteration. For $\tau_t = 25$, Fig. 8c shows that both DynVC-EDA-P and DynVC-EDA-M algorithms perform better than competitive algorithms.

Results related to benchmarks GTA2m and GTA2a are presented in Figs. 9 and 10, respectively. For both Figs. 9b and 10b DynVC-EDA-P outperforms the compared algorithms using $\tau_t = 5$. As shown in Figs. 9c and 10c, with a larger frequency of change $\tau_t = 25$, the DynVC-EDA-P algorithm performs better than DynVC-EDA-M in the first iterations but after a number of iterations (for example 300 iterations depicted in Fig. 10c), DynVC-EDA-M gets approximately the same result as DynVC-EDA-P because the used memory in the DynVC-EDA-M has more diverse results.

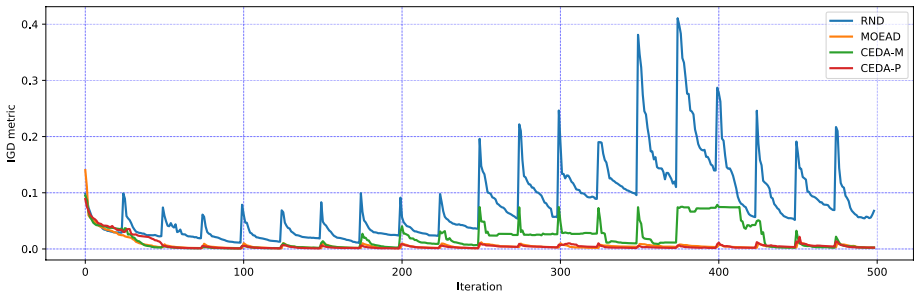
Figures 11 and 12 present the results for the benchmark GTA5 both addition (GTA5a) and multiplication (GTA5m) variants. These results show that the DynVC-EDA-P and DynVC-EDA-M algorithms achieve better results compared to the state-of-the-art algorithms, and they can adapt in an efficient way to the problem changes compared to the other algorithms.



(a) $n_t = 1$, $\tau_t = 10$



(b) $n_t = 10$, $\tau_t = 5$



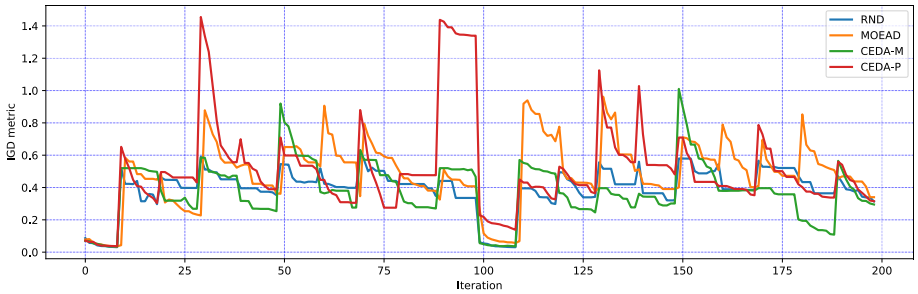
(c) $n_t = 10$, $\tau_t = 25$

Fig. 11 IGD over iteration for the GTA5a benchmark with different n_t and τ values

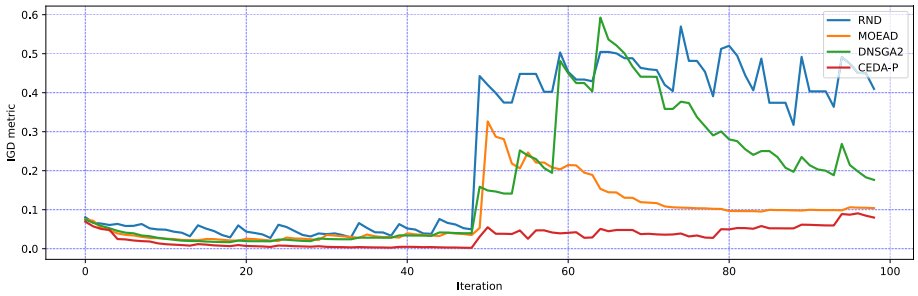
Figures 13 and 14 present the results for the GTA6 benchmark. In Figs. 13c and 14b we show the results using $n_t = 1$ and $\tau_t = 5$, respectively. The results obtained for both benchmarks show that the proposed DynVC-EDA-P maintains a good result each time the problem change. Contrary to the compared algorithms, where they start with a good approximation of the PS, but starting from iteration 50, the compared algorithms cannot maintain a good stability of the IGD metric.

Figures 7 and 8 show the same behavior, for $n_t = 1$ we see many variations of the IGD over iterations for all compared algorithms. However, the DynVC-EDA-M algorithm starts with the same results as the other algorithms, then during the time it gets better, as shown in Figs. 7a and 8a. The use of memory can explain this behavior. Every time our algorithm saves more solutions, this will help the vine-copula created from the memory to produce more diverse solutions, so the approximated PF will be more close to the Pareto Optimal Front. Figures 7b, c and 8b, c show the results for the rest of the experiments for GTA1a and GTA1m, the proposal as shown, get more stability of the IGD metric compared to the state-of-art algorithms.

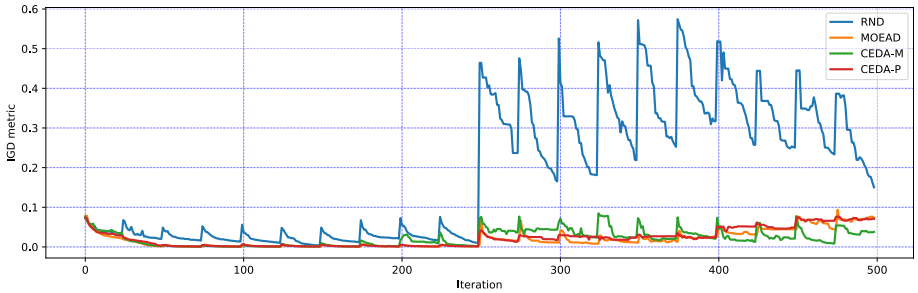
In the case of the GTA6a with $n_t = 1$, we can see that our algorithm has not a good stability, in terms of the IGD value,



(a) $n_t = 1$, $\tau_t = 10$



(b) $n_t = 10$, $\tau_t = 5$



(c) $n_t = 10$, $\tau_t = 25$

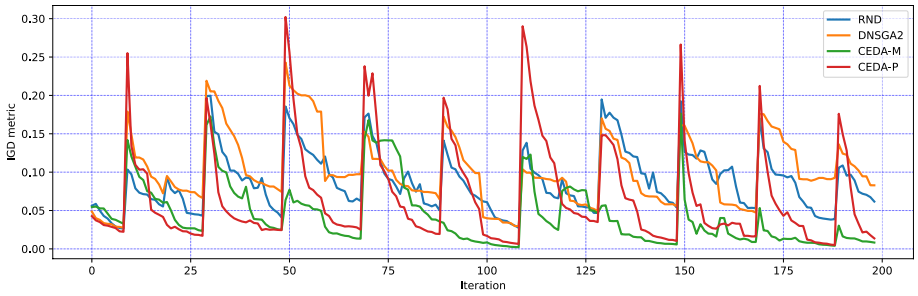
Fig. 12 IGD over iteration for the GTA5m benchmark with different n_t and τ values

the same as other algorithms. We can explain this behavior by the fact that the severity of the change is significant. Another argument can be that the GTA6m problem has a mixed modality where our algorithm cannot perform well as already explained from the results in Table 4.

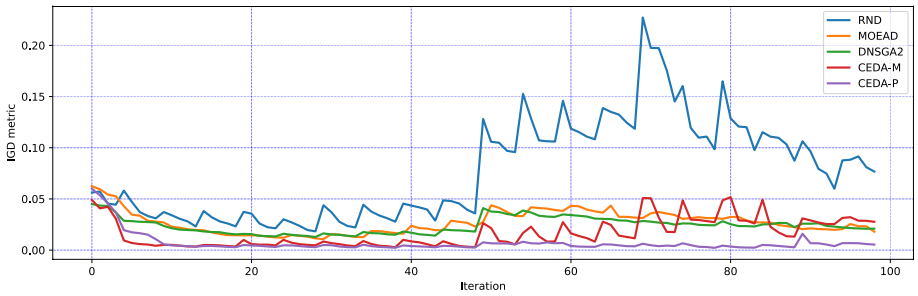
6 Conclusions

In this work, we have proposed an estimation of distribution algorithm to deal with the multiobjective problem that changes over time. There exist many methods to find a good approximation of the optimal solutions of the dynamic multiobjective problems. Some methods use memory to store the best solutions and re-use them when a change occurs in the problem, and others try to predict the up-coming optimal solutions according to the change history. Our proposal uses the power of the EDA to deal with the problem changes in two different ways the memory-based or the prediction-based one.

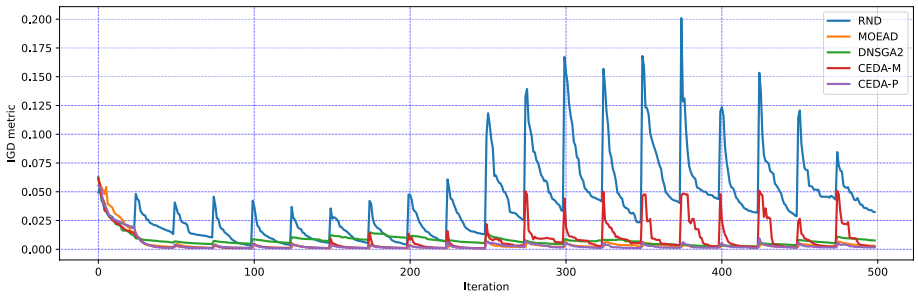
For the two methods, we use the well-known copula to estimate the distribution of the best solutions found so far and to get more advantages of the multi-variate copula. We model them using the vine-copula method. Indeed,



(a) $n_t = 1$, $\tau_t = 10$



(b) $n_t = 10$, $\tau_t = 5$

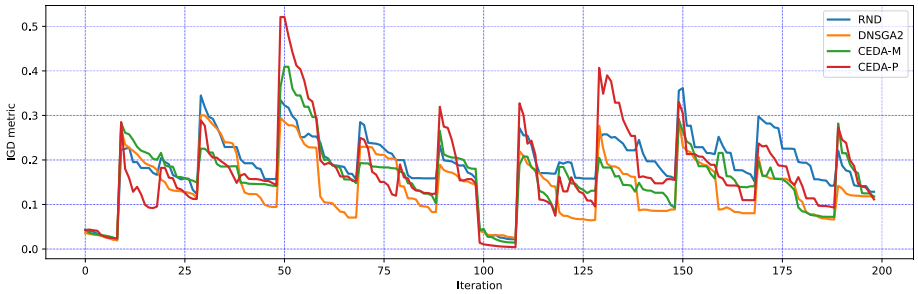


(c) $n_t = 10$, $\tau_t = 25$

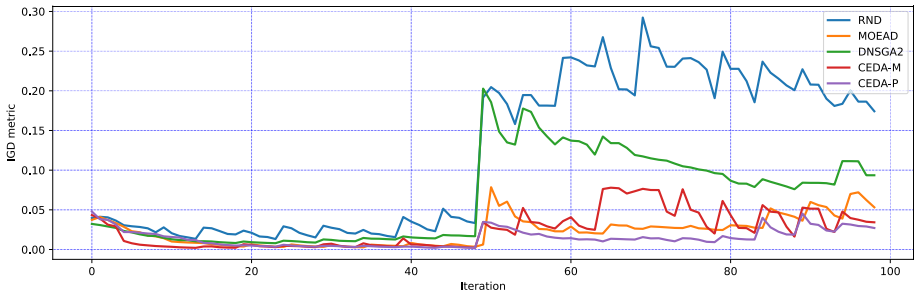
Fig. 13 IGD over iteration for the GTA6a benchmark with different n_t and τ values

vine-copula proved a good efficiency when dealing with a high dimension of the modeled solution compared with other methods. After estimating the best solutions using vine-copula, and when a change occurs, we produce new solutions using the created model.

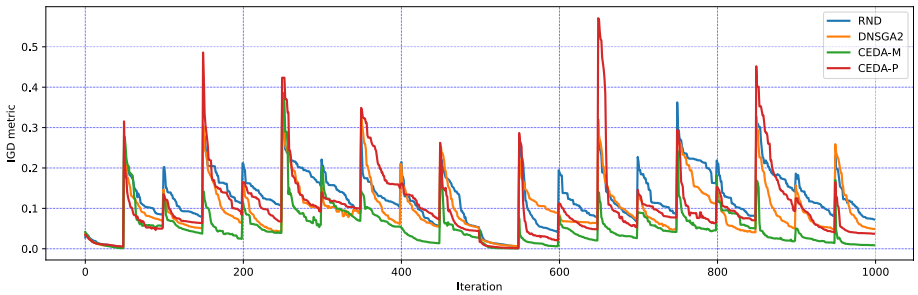
To evaluate the idea behind this work, we have tested it using many benchmarks. We tested it on the CEC2015 benchmarks, FDA problems, and the GTA benchmarks. Then, we calculate the quality of the obtained results in various parameter configurations using the MIGD metric known in dynamic multiobjective optimization. The obtained results show good qualities compared with state-of-art algorithms. Future work can be the use of the vine-copula model in the class of real-world dynamic multiobjective optimization problems and combinatorial problems.



(a) $n_t = 1$, $\tau_t = 10$



(b) $n_t = 10$, $\tau_t = 5$



(c) $n_t = 1$, $\tau_t = 50$

Fig. 14 IGD over iteration for the GTA6m benchmark with different n_t and τ values

Acknowledgements I want to express my very great appreciation to Professor Roberto Santana for his valuable and constructive suggestions during the planning and development of this research paper. His willingness to give his time so generously has been very much appreciated.

References

1. Aas K, Czado C, Frigessi A, Bakken H (2009) Pair-copula constructions of multiple dependence. Insur Math Econ 44(2):182–198
2. Abualigah LM, Khader AT (2017) Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. J Supercomput 73(11):4773–4795. https://doi.org/10.1007/s11227-017-2046-2
3. Abualigah LMQ (2019) Feature selection and enhanced krill herd algorithm for text document clustering. Springer, Berlin
4. Abualigah LMQ, Hanandeh ES (2015) Applying genetic algorithms to information retrieval using vector space model. Int J Comput Sci Eng Appl 5(1):19
5. Al-Sahaf H, Bi Y, Chen Q, Lensen A, Mei Y, Sun Y, Tran B, Xue B, Zhang M (2019) A survey on evolutionary machine learning. J R Soc N Zeal 49(2):205–228. https://doi.org/10.1080/03036758.2019.1609052
6. Azzouz R, Bechikh S, Ben Said L (2017) Dynamic multi-objective optimization using evolutionary algorithms: a survey. Springer, Cham, pp 31–70
7. Azzouz R, Bechikh S, Said LB (2017) A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy. Soft Comput 21(4):885–906
8. Bedford T, Cooke RM (2002) Vines-a new graphical model for dependent random variables. Ann Stat 30(4):1031–1068. https://doi.org/10.1214/aos/1031689016
9. Cheriet A, Cherif F (2015) A posteriori Pareto front diversification using a copula-based estimation of distribution algorithm. Int J Adv Comput Sci Appl (IJACSA) 6:13

10. Cheriet A, Cherif F, Taleb-Ahmed A (2016) Fast solutions enhancing using a copula-based EDA and SVM for many-objective problems. IFAC-PapersOnLine 49(12):781–786
11. Cheriet A, Santana R (2018) Modeling dependencies between decision variables and objectives with copula models. In: Proceedings of the genetic and evolutionary computation conference companion, GECCO '18. ACM, New York, NY, USA, pp 175–176. https://doi.org/10.1145/3205651.3205694
12. Cheriet A, Santana R (2019) Optimizing permutation-based problems with a discrete vine-copula as a model for eda. In: Proceedings of the genetic and evolutionary computation conference companion. ACM, pp 143–144
13. Cherubini U, Luciano E, Vecchiato W (2004) Copula methods in finance. Wiley, New York
14. Coello CAC, Van Veldhuizen DA, Lamont GB (2002) Evolutionary algorithms for solving multi-objective problems, vol 242. Springer, Berlin
15. Cuesta-Infante A, Santana R, Hidalgo JI, Bielza C, Larrañaga P (2010) Bivariate empirical and n-variate Archimedean copulas in estimation of distribution algorithms. In: Proceedings of the 2010 congress on evolutionary computation CEC-2010. IEEE, Barcelone, Spain, pp 1–8. https://doi.org/10.1109/CEC.2010.5586557
16. Czado C (2010) Pair-copula constructions of multivariate copulas. Copula theory and its applications. Springer, Berlin, pp 93–109
17. Deb K, Karthik S et al (2007) Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling. In: Evolutionary multi-criterion optimization. Springer, pp 803–817
18. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
19. Devroye L (1986) Sample-based non-uniform random variate generation. In: Proceedings of the 18th conference on winter simulation, WSC '86. ACM, New York, NY, USA, pp 260–265. https://doi.org/10.1145/318242.318443
20. Etxeberria R, Larranaga P (1999) Global optimization using Bayesian networks. In: Second symposium on artificial intelligence (CIMAF-99). Habana, Cuba, pp 332–339
21. Farina M, Deb K, Amato P (2004) Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Trans Evol Comput 8(5):425–442
22. Gao Y (2009) Multivariate estimation of distribution algorithm with Laplace transform Archimedean copula. In: International conference on information engineering and computer science, 2009. ICIECS 2009. IEEE, pp 1–5
23. Gao Y, Hu X, Liu H (2010) Estimation of distribution algorithm based on multivariate Gaussian copulas. In: 2010 IEEE International conference on progress in informatics and computing (PIC), vol 1. IEEE, pp 254–257
24. Gee SB, Tan KC, Abbass HA (2017) A benchmark test suite for dynamic evolutionary multiobjective optimization. IEEE Trans Cybern 47(2):461–472. https://doi.org/10.1109/TCYB.2016.2519450
25. Genest C, Rémillard B, Beaudoin D (2009) Goodness-of-fit tests for copulas: a review and a power study. Insur Math Econ 44(2):199–213
26. Genest C, Rivest LP (1993) Statistical inference procedures for bivariate archimedean copulas. J Am Stat Assoc 88(423):1034–1043. https://doi.org/10.1080/01621459.1993.10476372
27. Goh CK, Tan KC (2008) A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Trans Evol Comput 13(1):103–127
28. González-Fernández Y, Soto M (2012) A survey of estimation of distribution algorithms based on copulas. Technical Report, 2012, technical Report ICIMAF p 679
29. Hauschild M, Pelikan M (2011) An introduction and survey of estimation of distribution algorithms. Swarm Evol Comput 1(3):111–128
30. Helbig M, Engelbrecht A (2015) Benchmark functions for cec, special session and competition on dynamic multi-objective optimization. Technical report
31. Helbig M, Engelbrecht AP (2013) Benchmarks for dynamic multi-objective optimisation. In: 2013 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE). IEEE, pp 84–91
32. Helbig M, Engelbrecht AP (2013) Issues with performance measures for dynamic multi-objective optimisation. In: 2013 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE). IEEE, pp 17–24
33. Helbig M, Engelbrecht AP (2013) Performance measures for dynamic multi-objective optimisation algorithms. Inf Sci 250:61–81
34. Hyrš M, Schwarz J (2014) Multivariate Gaussian copula in estimation of distribution algorithm with model migration. In: 2014 IEEE symposium on foundations of computational intelligence (FOCI). IEEE, pp 114–119
35. Joe H (2014) Dependence modeling with copulas. CRC Press, Boca Raton
36. Kim G, Silvapulle MJ, Silvapulle P (2007) Comparison of semiparametric and parametric methods for estimating copulas. Comput Stat Data Anal 51(6):2836–2850
37. Larranaga P, Lozano JA (2002) Estimation of distribution algorithms: a new tool for evolutionary computation, vol 2. Springer, Berlin
38. Martins MS, Delgado M, Lüders R, Santana R, GonçSalves RA, de Almeida CP (2018) Exploring the probabilistic graphic model of a hybrid multi-objective Bayesian estimation of distribution algorithm. Appl Soft Comput 73:328–343
39. Martins MSR, Yafrani ME, Santana R, Delgado M, Lüders R, Ahiod B (2018) On the performance of multi-objective estimation of distribution algorithms for combinatorial problems. In: 2018 IEEE congress on evolutionary computation (CEC), pp 1–8. https://doi.org/10.1109/CEC.2018.8477970
40. Muruganantham A, Tan KC, Vadakkepat P (2016) Solving the IEEE CEC 2015 dynamic benchmark problems using Kalman filter based dynamic multiobjective evolutionary algorithm. In: Lavangnananda K, Phon-Amnuaisuk S, Engchuan W, Chan JH (eds) Intelligent and evolutionary systems. Springer, Cham, pp 239–252
41. Nelsen RB (2013) An introduction to copulas, vol 139. Springer, Berlin
42. Nguyen TT, Yang S, Branke J (2012) Evolutionary dynamic optimization: a survey of the state of the art. Swarm Evol Comput 6:1–24
43. Pelikan M, Sastry K, Goldberg DE (2006) Multiobjective estimation of distribution algorithms. In: Pelikan M, Sastry K, Cantú-Pa E (eds) Scalable optimization via probabilistic modeling. Springer, Berlin, pp 223–248
44. Salinas-Gutiérrez R, Hernández-Aguirre A, Villa-Diharce ER (2009) Using copulas in estimation of distribution algorithms. In: MICAI 2009: advances in artificial intelligence. Springer, pp 658–668
45. Salinas-Gutiérrez R, Hernández-Aguirre A, Villa-Diharce ER (2010) 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. ACM, pp 359–366
46. Salinas-Gutiérrez R, Hernández-Aguirre A, Villa-Diharce ER (2013) Incorporating regular vines in estimation of distribution algorithms. In: EVOLVE-A Bridge between Probability, set

oriented numerics and evolutionary computation. Springer, pp 91–121
47. Sastry K, Goldberg DE, Pelikan M (2005) Limits of scalability of multiobjective estimation of distribution algorithms. In: The 2005 IEEE congress on evolutionary computation, 2005, vol 3. IEEE, pp 2217–2224
48. Shakya S, McCall J (2007) Optimization by estimation of distribution with deum framework based on markov random fields. Int J Autom Comput 4(3):262–272
49. Sheather SJ, Jones MC (1991) A reliable data-based bandwidth selection method for kernel density estimation. J R Stat Soc Ser B (Methodol) 53:683–690
50. Sklar A (1973) Random variables, distribution functions, and copulas. Kybernetica 28:449–460
51. Slowik A, Kwasnicka H (2020) Evolutionary algorithms and their applications to engineering problems. Neural Comput Appl 32(16):12363–12379. https://doi.org/10.1007/s00521-020-04832-8
52. Soto M, Gonzalez-Fernandez Y (2010) Vine estimation of distribution algorithm. Technical report. 2010-561, ICIMAF
53. Soto M, Gonzalez-Fernandez Y, Ochoa A (2012) Modeling with copulas and vines in estimation of distribution algorithms. arXiv preprint arXiv:1210.5500
54. Soto M, Ochoa A, Arderí RJ (2007) Gaussian copula estimation of distribution algorithm. Technical report. 2007-406, ICIMAF
55. Soto MR, Ochoa A, González-Fernández Y, Milanés Y, Álvarez A, Carrera D, Moreno E (2012) Vine estimation of distribution algorithms with application to molecular docking. In: Shakya S, Santana R (eds) Markov networks in evolutionary computation. Springer, Berlin, pp 175–190
56. Tsukahara H (2005) Semiparametric estimation in copula models. Can J Stat 33(3):357–375
57. Wang L, Guo X, Zeng J, Hong Y (2010) Using Gumbel copula and empirical marginal distribution in estimation of distribution algorithm. In: 2010 Third international workshop on advanced computational intelligence (IWACI). IEEE, pp 583–587
58. Wang LF, Zeng JC (2010) Estimation of distribution algorithm based on copula theory. In: Chen Y (ed) Exploitation of linkage learning in evolutionary algorithms. Springer, Berlin, pp 139–162
59. Wang LF, Zeng JC, Hong Y (2009) Estimation of distribution algorithm based on archimedean copulas. In: Proceedings of the first ACM/SIGEVO summit on genetic and evolutionary computation, GEC '09. Association for Computing Machinery, New York, NY, USA, pp 993–996. https://doi.org/10.1145/1543834.1543991
60. Wang Y, Li B (2010) Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. Memet Comput 2(1):3–24
61. Yan J et al (2007) Enjoy the joy of copulas: with a package copula. J Stat Softw 21(i04):1–21
62. Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
63. Zhang Q, Liu W, Li H (2009) The performance of a new version of moea/d on CEC09 unconstrained mop test instances. IEEE Cong Evol Compt 1:203–208
64. Zhang Q, Yang S, Jiang S, Wang R, Li X (2019) Novel prediction strategies for dynamic multiobjective optimization. IEEE Trans Evol Comput 24(2):260–274
65. Zhang Q, Zhou A, Jin Y (2008) RM-MEDA: a regularity model-based multiobjective estimation of distribution algorithm. IEEE Trans Evol Comput 12(1):41–63
66. Zhou A, Jin Y, Zhang Q (2014) A population prediction strategy for evolutionary dynamic multiobjective optimization. IEEE Trans Cybern 44(1):40–53
67. Zhou A, Qu BY, Li H, Zhao SZ, Suganthan PN, Zhang Q (2011) Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol Comput 1(1):32–49
68. Zitzler E, Laumanns M, Thiele L, Zitzler E, Zitzler E, Thiele L, Thiele L (2001) SPEA2: improving the strength pareto evolutionary algorithm
69. Zou F, Yen GG, Tang L, Wang C (2021) A reinforcement learning approach for dynamic multi-objective optimization. Inf Sci 546:815–834. https://doi.org/10.1016/j.ins.2020.08.101