Dynamic Multi-objective Estimation of Distribution Algorithm Based on Domain Adaptation and Nonparametric Estimation

Article *in* Information Sciences · January 2018

DOI: 10.1016/j.ins.2017.12.058

CITATIONS
0

READS
36

4 authors, including:

Min Jiang
Xiamen University
63 PUBLICATIONS **156** CITATIONS
SEE PROFILE

Qiu Liming
Xiamen University
4 PUBLICATIONS **1** CITATION
SEE PROFILE

Zhongqiang Huang
Xiamen University
11 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Dynamic Multi-objective Estimation of Distribution Algorithm Based on Domain Adaptation and Nonparametric Estimation

Min JIANG[a,b], Liming QIU[a,b], Zhongqiang HUANG[a,b], Gary G. YEN[*,c]

[a]*Department of Cognitive Science and Technology, Xiamen University, China*
[b]*Fujian Key Lab. of Brain-like Computing and Application, Xiamen University, China*
[c]*School of Electrical and Computer Engineering, Oklahoma State University, USA*

Abstract

Although dynamic optimization and multi-objective optimization have made considerable progress individually, solving dynamic multi-objective optimization problems remains a monumental challenge since their multiple, conflicting objectives could change over time. In this paper, we propose a Domain Adaptation and Nonparametric Estimation-based Estimation of Distribution Algorithm, called DANE-EDA, to solve dynamic multi-objective optimization problems. Notable features of the proposed algorithm include the importance sampling, nonparametric density estimation, probabilistic prediction, and a domain adaptation technique seamlessly unified under an innovative framework. The design takes full advantage of the powerful Monte-Carlo method and transfer learning technique. This kind of combination will help the proposed algorithm to maintain a delicate exploration-exploitation trade-off from temporal and spatial perspectives. At the same time, it will help the proposed algorithm to overcome the shortcomings caused by transfer learning, specifically, the loss of the diversity. After proving convergence and analyzing the computational complexity of the DANE-EDA, we compare the proposed method with nine EDAs or dynamic multi-objective optimization algorithms on twelve different test instances. The experimental results affirm the effectiveness of the proposed method in addressing dynamic multi-objective optimization problems.

Key words:
Dynamic multi-objective optimization, Domain adaption, Nonparametric statistics, Importance Sampling, Transfer learning

1. Introduction

Many real-world problems not only need to optimize multiple competing objectives simultaneously, but also are characterized by time-varying objective functions or dynamic environments. These problems are called Dynamic Multi-objective Optimization Problems (DMOPs). A typical scenario where DMOPs arises comes from job shop scheduling. In these problems, new job demands may arrive or machines may break down or wear out during operation [6]. Therefore, the job schedules should be dynamically modified to accommodate conflict, while meeting the multiple objectives of maximizing the yield and minimizing material waste. Another convincing example is dynamic portfolio management, which is a common optimization problem in modern finance [28]. This problem aims to obtain an optimal allocation of assets to maximize profit, while minimizing risk of investment. In engineering, dynamic portfolio optimization problems are common in deregulated electricity markets in which the operations of different power stations are controlled and coordinated to maximize profit and minimize risk. Obviously, a good solution of the DMOPs often leads to significant economic benefits.

Although multi-objective optimization and dynamic optimization (mainly for single-objective) have separately made much progress, solving a given DMOP remains a challenging research topic, especially by means of analytic approaches. Therefore designing different heuristic algorithms remains appealing to the scientific community [34]. As a meta-heuristic approach, the Estimation of Distribution Algorithm (EDA) [25] combines statistical machine learning with a population-based evolution heuristic. Most EDAs usually establish a probabilistic model using statistical machine learning methods and then use this model to generate the next population. Different from traditional evolutionary algorithms which heavily rely upon proper designs of crossover and mutation operations, the learning and sampling procedures over probabilistic models in EDAs determine the effectiveness and efficiency of the performance. Many researchers [15] believe that the EDAs can provide unique advantages which other meta-heuristic methods do not have for solving optimization problems, particularly for handling high dimensionality or complex optimization tasks.

Significant progress has been made in the EDA field recently. For example, the multi-objective Bayesian Optimization Algorithm (mBOA), proposed by Khan *et al.* [21], combines the Bayesian Optimization Algorithm (BOA) [32] with the Non-dominated Sorting Genetic Algorithm (NSGA-II) for solving multi-objective optimization problems. In essence, this algorithm wants to achieve some kind of exploration-exploitation balance by imposing proper selection pressures, and the Bayesian network plays a crucial role in this process.

[*]Corresponding author, gyen@okstate.edu
Email address: minjiang@xmu.edu.cn (Min JIANG)

The multi-objective hierarchical BOA (mohBOA), proposed by Pelikan *et al.* [33], also adopts a similar procedure originated from NSGA-II to select possible candidate individuals. The unique design of mohBOA is that, after candidate individuals are selected, mohBOA finds the clusters of promising solutions by k-means clustering algorithm and uses restricted tournament replacement to merge the newly obtained clusters with the old population by using restricted tournament replacement to generate the next population. The experiments show that mohBOA can solve the onemax-zeromax and trap5-invtrap5 [33] in low-order polynomial time. Shah *et al.* [36] proved that ϵ-hBOA, an ϵ-dominance hierarchical BOA, outperforms two chosen evolutionary algorithms in solving multi-objective d-dimensional knapsack problems. However, most of the existing EDA methods have room for improvement when solving the dynamic multi-objective optimization problems.

In recent years, the prediction-based approach [29] has drawn more attention from the field of dynamic optimization. This class of methods predicts the changing optimization functions typically using some machine learning techniques, and then make a prediction of where it is more hopeful to find a "good" solution, such that the algorithms can accommodate the changes in advance. This is one of the reasons the prediction-based approaches can substantially improve performance of an algorithm handling the DMOPs. As expected, it will be a promising approach to enhance the performance of the dynamic estimation distribution algorithms by combining the EDA algorithms with a machine learning technology [20].

However, traditional machine learning methods are often based on an assumption that the samples used to build the model, and the samples to be tested by the model follow the same distribution, which is the so-called Independent Identically Distributed (IID) hypothesis. Under dynamic environments, this hypothesis is very difficult to be satisfied, so that is one of the major reasons why the existing prediction-based methods leave much room to be desired when the training samples and the predicted samples fail to meet the IID hypothesis. In many cases, a poor prediction model can improve nothing but causes more trouble to the host algorithm.

Transfer learning techniques [31] allows the distribution of data used in training and testing to be different and it is becoming a useful weapon to overcome this difficulty. So a combination of transfer learning with DMOP algorithm will be promising since that historical information about the Pareto-optimal Front (POF) or Pareto-optimal Set (POS) is very useful. The reason is that for a DMOP, the POSs or POFs at different times are correlated, but not exactly the same.

Unfortunately at the same time, this combination also causes some problems. One of the problems is that the samples are clustered in the latent space created by the transfer learning method [30], which leads to a poor diversity in the solutions of the corresponding multi-objective optimization problem.

In this paper, a Domain Adaptation and Nonparametric Estimation based EDA, called DANE-EDA, is proposed to solve dynamic multi-objective optimization problems. The motivation of the proposed algorithm comes in two main parts. In one hand, we want to preserve merits of the estimation distribution algorithms and the transfer leaning [18], which means DANE-EDA predicts the changing Pareto front via reusing past "knowledge". On the other hand, we strive to overcome the shortcomings caused by transfer learning, i.e., the loss of diversity in solutions by means of nonparametric statistic and importance sampling methods.

The main contribution of the proposed EDA rests on that the importance sampling, nonparametric density estimation, and domain adaptation are unified seamlessly under an innovative framework to solve dynamic multi-objective optimization problems. More specifically, this algorithm assigns each individual (solution) a sparseness value obtained by a domain adaptation method and a nonparametric probabilistic density estimation method, and then filters offspring out by using an importance sampling. The design takes full advantage of the powerful Monte-Carlo method and transfer learning technique, and this kind of combination will help the proposed algorithm to maintain a delicate exploration-exploitation trade-off from temporal and spatial perspectives. At the same time, it will help the proposed algorithm to overcome the shortcomings caused by transfer learning, i.e., the loss of the diversity.

We hope this process achieves the exploration-exploitation balance in both time and spatial dimensions, and two techniques are used to achieve this goal. At first, we use the domain adaptation approach to reuse knowledge at different points in time. On the other hand, we adapt an important sampling method, such that we can generate and select candidate solutions at different points in space, and that is the reason we say that the proposed method wants to keep the exploration-exploitation trade-off from temporal and spatial perspectives at the same time during searching the POF.

The rest of this paper is organized as follows. In Section 2, we briefly introduce background and review related studies. Then we provide a detailed description of the proposed method and theoretical analysis in Section 3 and Section 4. In Section 5 we describe evaluation standards, test instances and comparative methods. In Section 6, the experimental results and some discussions are presented. In the Section 7, we sum up the main work of this research and future research direction is also highlighted.

2. Preliminaries and Related Works

2.1. Concepts of Multi-objective Optimization

In this research, dynamic multi-objective optimization problems are considered, that is, their objective functions will change over time. Formally, a dynamic multi-objective optimization problem is defined as:

$$\textbf{Minimize } F(x,t) = \langle f_1(x,t), f_2(x,t), ..., f_M(x,t)\rangle$$

$$s.t.\ x \in \Omega,$$

where $x = \langle x_1, x_2, \ldots, x_n\rangle$ is the decision vector and t is the time or environment variable. $f_i(x,t) : \Omega \rightarrow \mathbb{R}$ $(i = 1, \ldots, M)$. $\Omega = [L_1, U_1] \times [L_2, U_2] \times \cdots \times [L_n, U_n]$. $L_i, U_i \in \mathbb{R}$ are the lower and upper bounds of the i-th decision variable, respectively.

Definition 1. [Dynamic Decision Vector Domination] *At time t , a decision vector x_1 Pareto dominate another vector x_2, denoted by $x_1 \succ_t x_2$, if and only if*:

$$\begin{cases} \forall i = 1, \ldots, M, & f_i(x_1, t) \leq f_i(x_2, t) \\ \exists i = 1, \ldots, M, & f_i(x_1, t) < f_i(x_2, t) \end{cases}. \quad (1)$$

Definition 2. [Dynamic Pareto-optimal Set] *Both x and x^* are decision vectors, and a decision vector x^* is said to be nondominated at time t if and only if there is no other decision vector x such that $x \succ_t x^*$ at time t. The Dynamic Pareto-Optimal Set (DPOS) is the set of all Pareto optimal solutions at time t, that is*:

$$DPOS = \{x^* | \nexists x,\ x \succ_t x^*\}.$$

Definition 3. [Dynamic Pareto-optimal Front] *At time t, the Dynamic Pareto-Optimal Front (DPOF) is the corresponding objective vectors of the DPOS.*

$$DPOF = \{F(x^*, t) | x^* \in DPOS\}.$$

Remark 1. *If time factor has not been considered, the DPOS and DPOF will be degenerated to Pareto-optimal Set (POS) and Pareto-optimal Front (POF), respectively .*

Please note that in the remainder of this paper, we will not intentionally make a clear distinction between DPOS and POS.

2.2. Estimation of Distribution Algorithms

The main feature of EDAs [24] is that they incorporate probabilistic modeling into the evolutionary framework for optimization. Instead of individual-based evolution strategy, EDAs use a population-based strategy to evolve the whole population and explore the search space through building and sampling probabilistic models of promising candidate solutions.

Compared to traditional Evolutionary Algorithms (EAs), EDAs possess some unique advantages: 1) they provide explicit probabilistic models, so these models can be analyzed mathematically; 2) a prior knowledge can be incorporated into the models in a more principled way; 3) instead of managing a population through variations, reducing memory requirement in EDAs is highly expected.

Various EDA algorithms have been developed and there are significant differences among them. However, a majority of these designs would generally involve the following two steps:

1. Firstly, a probabilistic model is built for modelling the solution space, usually through the process of population evaluating, candidate selection, and statistical machine learning. This step is called as *Model Building*.
2. Based on the constructed model, some kind of sampling method is used to generate a new population. We call this step as *Sampling*.

2.3. Related Works

In recent years, DMOPs have drawn lots of attention from researchers [3] in the optimization field, and most existing algorithms can be roughly classified into the following categories: Increasing/Maintaining Diversity methods, Multi-population based methods, Memory based methods, and Prediction based methods.

The increasing diversity methods tend to add variety to the population by using a certain type of methodology when the environment change was detected. For example, Cobb *et al.* proposed the triggered hypermutation method [8], and the basic idea of this method is that when change is identified, the mutation rate would be increased immediately, and this would make the converged population divergent again. This approach leaves some room for improvements, and one of them is that the mutation rate is in a state of uncontrolled change during the whole process, and this ultimately results in reduced performance of the algorithm. Therefore, Vavak *et al.* [41] presented a mutation operator, called variable local search (VLS), to address the problem.

The strategy that the VLS adopted was to gradually increase the mutation rate. Yen *et al.* [43] proposed a dynamic EA which relocates the individuals based on their change in function value due to the change in the environment and the average sensitivities of their decision variables to the corresponding change in the objective space. This approach can avoid the drawbacks of previous methods to a certain extent.

Maintaining diversity is also one of the effective means which can help the algorithm track the changing optimum as soon as possible. Dynamic NSGA-II (DNSGA-II) [11], proposed by Deb *et al.*, handles the DMOPs by introducing diversity when change is detected. There are two versions of the proposed DNSGA-II and they are respectively known as DNSGA-II-A and DNSGA-II-B. In the DNSGA-II-A, the population is replaced by some individuals with new randomly created solutions, while in the DNSGA-II-B, diversity was guarded by replacing a percentage of the population with mutated solutions.

Grefenstette [14] proposed a Random Immigrants Genetic Algorithm (RIGA) also shares a similar idea, and the method replaces some individuals in the population randomly. The idea of the RIGA is that introducing new genetic materials into the population can avoid the whole population converging toward a small area in the process of evolution. However, the drawback of the primitive immigrant method was the fitness values of the introduced individuals were usually low, so large amounts were eliminated during the selection stage, and as a result, it was very difficult to introduce different genes into the population.

Memory mechanism enables EAs to record past information, and when they detect changes have occurred, stored information can be reused to improve the performance of the algorithm. Existing research showed that memory-based approaches tend to be more effective on the DMOPs with periodically changing environments. Branke [5] proposed a direct memory scheme where the best individuals in the population will be saved in an archive, and when the algorithm detects a change, those saved individuals can be retrieved and returned to the population to replace the same number of individuals.

On this basis, the different approaches were presented. In [13], the authors proposed a co-evolutionary multiobjective algorithm which hybridizes competitive and cooperative mechanisms to solve the DMOPs. In this algorithm, the out-of-date archived solutions are replaced by an external population. In [42], the authors presented an algorithm called MS-MOEA to tackle the challenges of DMOPs. In the method, adaptive genetic and differential operators were used to speed up the convergence speed and a Gaussian local search operator was employed to prevent from premature convergence.

The above methods meet some problems when the environment changes. As a result the authors in [2] proposed an adaptive hybrid population management strategy using memory, local search and random strategies to effectively handle environment dynamicity in DMOPs. The special feature of this algorithm is that it can adjust the number of memory and random solutions to be used according to the change severity. In [19], the authors proposed a method called called Steady-state and Generational Evolutionary Algorithm (SGEA), and SGEA detects and reacts to changes in a steady state manner. When a change is detected, SGEA reuses a portion of old solutions with good diversity and re-evaluates them.

The Multi-population strategy is considered as one efficient solution for the DMOPs, especially for the multiple peaks and the competing peaks problems. Branke *et al.* [7] proposed the self-organizing scouts method, and this method splits the population into scout and base populations, and the two populations are responsible for exploitation and exploration respectively. In other words, the base population searches for the optimal solution and if the base population finds a peak, then the scout population is generated to track the change of this new peak.

Li and Yang [26] used a multi-population particle swarm optimization (PSO) algorithm to solve multiple peaks problems. In their method, a population uses evolutionary programming, which shows a better global search ability when compared to other EAs, to explore the most hopeful areas in the whole search space, and at the same time, several subpopulations use the fast PSO algorithm to find the local optima. Yang [44] used the hierarchical clustering technique to divide the population into different subpopulations, and the main advantage of this design is that the initial individuals of the subpopulations can be generated automatically according to the fitness landscape.

A good dynamic optimization algorithm must be able to track the changing optimal solution even under high severity and frequency changes. One of the ways is to reuse information from previous generations to speedup the optimization search. So, the prediction-based DMOPs algorithms have drawn lots of attention. Stroud [40] proposed the Kalman-extended Genetic Algorithm (KGA), and the basic idea of the KGA was that two types of uncertainties surrounded the estimated value of an individual in a dynamical environment. The first type of uncertainty is produced by the dynamic of the environment while the second type was related to the evaluation of individuals. For the different situations, the KGA has two different ways to update the covariances, and uses the Kalman filter technique to predict the two uncertainties which allows the algorithm to work well in a dynamic environment.

Bosman [4] considered that a decision made at one point would affect the optima obtained in the future, so for the dynamic optimization problems, he proposed an algorithmical framework which integrated machine learning, statistical learning, and evolutionary computation, and this framework can effectively predict what the state of environment is going to be. In [35], the authors suggested that the state of an optimum should contain the location and the speed information, so the Kalman filter technique can be used to estimate the state of the system and the error. The authors proposed an EA to measure the state of the past optimum and then use the Kalman filter to obtain an estimated value of the optimum in the next time instance.

In [47], Zhou *et al.* presented an algorithm, called Population Prediction Strategy (PPS), to predict a whole population instead of predicting some isolated points. There are two key concepts in the research: center point and manifold. Whenever a change is detected, the algorithm uses a sequence of center points obtained from the search progress to predict the next center point, and at the same time, the previous manifolds are used to estimate the next manifold. The main problem of this method is that, it is difficult to obtain historical information at the beginning stage, and this may lead to poor convergence.

In [29], the authors proposed a dynamic multiobjective evolutionary algorithm, MOEA\D-KF, which used a Kalman filter (KF) to predict the moving optima in the decision space. The novelty of the approach is that a scoring scheme is designed to hybridize the KF prediction with a random reinitialization method. The predictions help to guide the search toward the changed optima, thereby accelerating convergence.

In [50], the authors proposed a prediction strategy based on center points and knee points (CKPS) consisting of three mechanisms for solving dynamic multiobjective optimization problems. The first mechanism is a method of predicting the nondominated set based on the forward-looking center points, and the second one is to introduce the knee point set to predict accurately the location and distribution of the Pareto front after an environmental change. Finally, an adaptive diversity maintenance strategy was proposed for generating some random individuals of the corresponding number according to the degree of difficulty of the problem to maintain the diversity of the population. In [1], the authors proposed a continuous genetic algorithm to solve the second-order boundary value problems where smooth solution curves are used throughout the evolution of the algorithm.

In [27], a modified co-evolutionary multi-swarm particle swarm optimizer is proposed to solve dynamic multi-objective optimization problem. In the proposed framework, the number of swarms (PSO) is determined by the number of the objective functions, and all of these swarms utilize an information sharing strategy to evolve cooperatively. At the same time, the authors developed a velocity update rule and an effective boundary constraint technique during evolution of each swarm. Finally, a similarity detection operator is used to detect whether a change has occurred, followed by a memory based dynamic mechanism to response to the change.

The EDA field yields remarkable results and we just review the researches related to our work. Briefly speaking, estima-

tion of distribution algorithms are stochastic optimization techniques which explore the space of potential solutions via probabilistic models of candidate solutions. MBN-EDA, in short for Multi-dimensional Bayesian Network based Estimation of Distribution Algorithm, proposed by Karshenas [20] builds a joint model, multidimensional Bayesian network of variables and objectives. Both the variables and the objectives are encoded in this model. MBN-EDA not only captures the dependencies between variables, but also between objectives and variables. Gaussian Bayesian networks and regularization techniques are employed in MBN-EDA to obtain a more robust estimation and a sparse structure.

It is well known that the Pareto set of a continuous MOPs is a piecewise continuous $m-1$ dimensional manifold. Based on this property, Zhang *et.al* proposed a Regularity Model-based Multi-objective Estimation of Distribution Algorithm, RM-MEDA [45], which uses local Principal Component Analysis (PCA) to build a model in $m-1$ dimensional manifold, where m is the number of objectives. In this algorithm, the population is partitioned into several disjoint clusters using local PCA and the manifold is estimated upon these. However, the RM-MEDA does not use the location information of individuals and may easily fail in the problems with many local Pareto fronts.

Non-parametric probabilistic models also have received great interest among the researchers. For example, Multi-Objective Parzen-based EDA (MOPEDA) [10] applies a Parzen estimator, a non-parametric method, to approximate the probability density of solutions on the Pareto front. In this approach, a mixture of uni-modal kernel functions is learned by Parzen estimator, and both Gaussian and Cauchy distributions are adopted alternatively during evolution.

In some situations, it is very difficult to judge whether a solution is an optimal solution or not. However, it is relatively easy to discern a dominated solution from those who are not. Generating-filtering strategy offers effective assistance to respond to the situations. The basic idea behinds the strategy is when the impossible options are removed, the difficulty of solution-finding becomes far easier. The EDAs based on this strategy [48] usually emphasize the cooperation between population generating and population filtering, and avoid building a model explicitly or using a predefined model, so in this sense an algorithm based on the Generating-filtering strategy will have advantage over traditional EDAs.

However, the generating-filtering strategy alone remains very difficult to solve the dynamic multi-objective optimization problems effectively. The reason is that the dynamic multi-objective optimization problem not only needs to search in a large solution space, but also because the optimization objectives are changing constantly. The proposed DANE-EDA not only works with the generating-filtering strategy [48], it also combines the nonparametric density estimation and the importance sampling together, such that DANE-EDA can assume the advantages from both Monte-Carlo methods and nonparametric statistical methods.

It is worth noting that some EDA-based methods are also promising for solving DMOPs. For example, we proposed an EDA [16] which constructs a probabilistic model to predict the trend of the best solutions, so as to promote the search speed for finding the POF. In this approach, one of the purposes is to achieve the diversity of population from the spatial point of view. The research can be regarded as the initial point of the work presented here. The trend prediction model is taken as the static part of the proposed algorithm, and that is to say the probabilistic model is used to generate new offspring. However, there are significant differences between the two approaches. At first, we use rigorous mathematical arguments to improve probabilistic models presented in [16], thus we can carry on property analysis for the proposed algorithm. What is more, the EDA-based prediction method is combined with a transfer learning method, such that the proposed method can remain both the advantages of EDA method and transfer learning approach, and the experimental results show that this combination provides significant performance gains for solving different benchmark functions.

It is obvious that transfer learning techniques will benefit the DMOP algorithms greatly since for a DMOP, the POSs or POFs at different times are correlated, but they are not be exactly the same. Therefore a prediction model built by a transfer learning method will be much better than the prediction model which is created by traditional machine learning methods. However, in another respect, we found that transfer learning, especially for the domain adaptation, often lead to the diminishing diversity in solutions. The proposed approach, DANE-EDA, not only can benefit from the advantages of transfer learning, but also overcome the disadvantages caused by the transfer learning.

3. The DANE-EDA

In the beginning of this section, the framework of DANE-EDA is depicted, and after that three components of the proposed algorithm are introduced in detail: nonparametric density estimation based importance sampling, the probabilistic inertial prediction model, and transfer learning based initial population generation.

3.1. The Algorithm Framework

A formal definition of a hyper-solution is as follows:

Definition 4. *For an optimization problem with M objective functions, a hyper-solution is a tuple $P = \langle x, d \rangle$, where x is an n-dim vector representing the solution of the multiobjective optimization problem and d is an n-dim unit vector representing the direction of the hyper-solution.*

The direction d of the hyper-solution can be used to update the probabilistic inertial prediction model, which will be described in Subsection 3.3, and this model will produce more good offspring which can approach the POF more quickly.

The proposed algorithm, the DANE-EDA, can be divided into two parts: the static part and the dynamic part. Fig. 1 and Fig. 2 are used to illustrate the two parts respectively. Please note that the DANE-EDA never stops since we assume that the

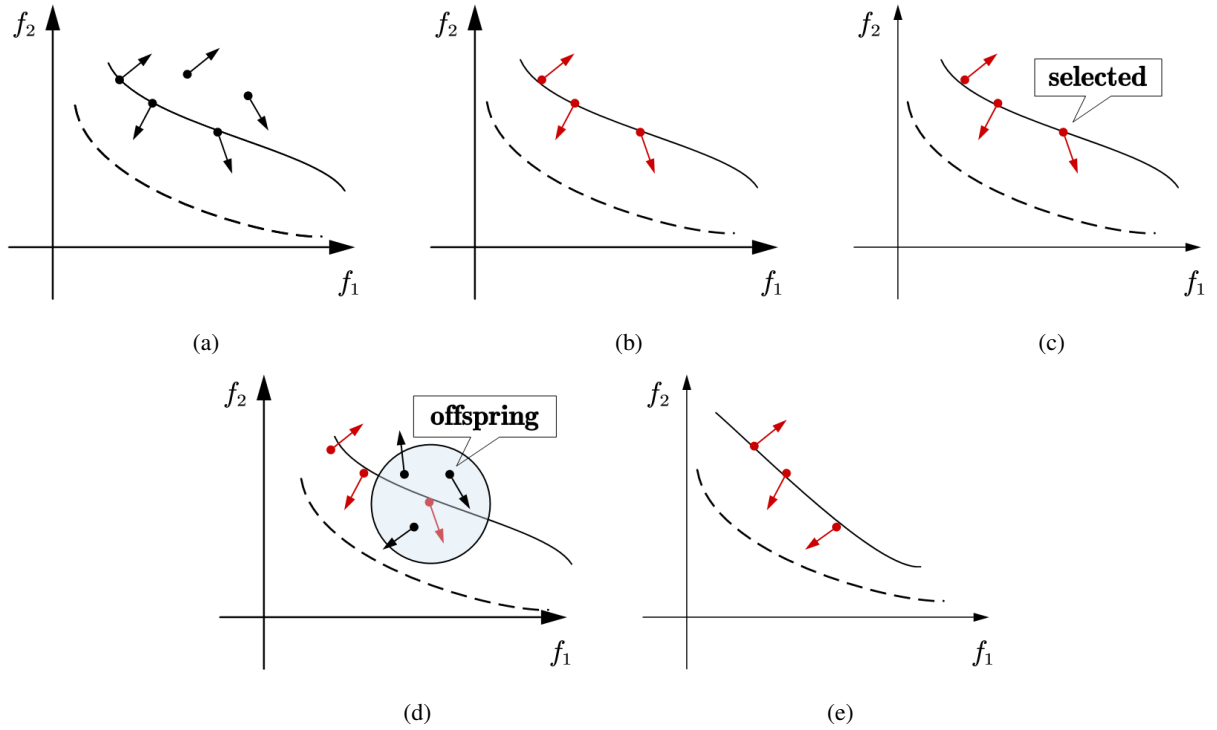

Figure 1: The static part of DANE-EDA. (a) To generate some solutions randomly. (b) To calculate the non-dominated set (c) To select an individual from the non-dominated set via important sampling. (d) To generate some new hyper-solutions around the individual just selected by using the probabilistic inertial model (e) To update the Pareto set and the Pareto front.

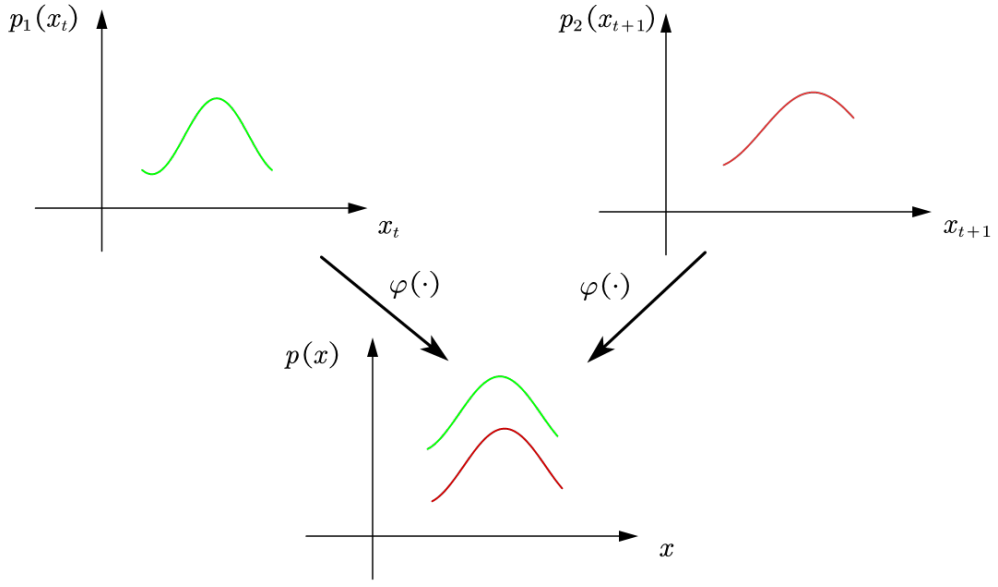

Figure 2: The dynamic part of DANE-EDA. This part maps the distributions at t (the upper-left) and $t+1$ (the upper-right) into a latent space, in which the two distributions will be as similar as possible.

dynamic optimization problem is constantly running in need of solutions.

Those operations in static part are time independent, and it can be regarded as one iteration when time is fixed. This part involves the following steps: At first, DANE-EDA generates some hyper-solutions (the black points in the Fig. 1(a)), and then it calculates the Non-dominated set [1] (the red points in the Fig. 1(b)). In the subsequent step, the DANE-EDA estimates the probability density of the distribution which corresponds to the Non-dominated set via a nonparametric kernel density estimation method, and then the proposed algorithm will select one hyper-solution by using important sampling (the Fig. 1(c)), and the principle is that the hyper-solution with high sparseness value can get more chances to be selected and vice versa. After this step, the DANE-EDA will feed the selected hyper-solution into the proposed prediction model to generate some new hyper-solutions (the black points in the Fig. 1(d)), and the next step will get updated Non-dominated set (Fig. 1(e)) according to the hyper-solutions. The program will loop continuously until the termination condition is satisfied.

At the next clock step, the environment will change and the objective functions will be different. At this moment, the DANE-EDA will use a technique, Transfer leaning based Initial Population Generator (Tr-IPG for short), to produce an initial population to search for the POS under the new environment. Fig. 2 illustrates the key ideas of this part. The upper-left is a distribution which the solutions at time t obey, and upper-right is a distribution that the solutions at time $t + 1$ follow. These two distributions come from the same optimization function, although the optimization functions will change over time, that is to say they are not exactly the same, but there is a correlation between the two distributions due to the existence of the common parts of the function at different moment. The Tr-IPG procedure can be used to map two distributions into a latent space, such that the two distributions are as similar as possible. Thus the POS obtained at the time t can be used to construct a population to search for the POS of the optimization function at the time $t + 1$.

What we want to point out is that one of the major procedures of the static part is to help the dynamic part to overcome the problem caused by transfer learning - the diminishing diversity in the solutions.

3.2. Nonparametric Density Estimation based Importance Sampling

The primary function of Nonparametric Estimation based Importance Sampling (NEIS) is to filter out "the less important hyper-solutions" with higher probability, such that the remaining hyper-solutions can be used to generate the offspring. During the filtering procedure, the principle we followed is "maintaining diversity" from temporal and spatial perspectives.

The basic idea of our method is to assign an importance weight, called sparseness, to every hyper-solution in the non-dominated set, and the weight can be considered as a probability value and it will change over time. The hyper-solutions with higher sparseness values will have a better chance of producing more offspring than the hyper-solutions with lower sparseness values.

In this research we employ kernel density estimation to calculate the sparseness value, and its definition is as follows:

Given a set of hyper-solutions $DP = \{p_1, \ldots, p_n\}$, where $p_i = \langle x_i, d_i \rangle, (1 \leq i \leq n)$ is a hyper-solution. The probability density of a hyper-solution p_k is calculated by kernel density estimation, that is:

$$Density\,(p_k) = \frac{1}{nh} \sum_{i=1}^{n} \kappa\left(\frac{x_k - x_i}{h}\right), \tag{2}$$

where $\kappa(\cdot, \cdot)$ is a kernel function [37], and Gaussian kernel function is used in this research. The parameter h adjusts the bandwidth of the probabilistic density curve. In other words, the probability density curve will be smooth when h is large, and conversely the curve is steep if the parameter is small.

The probability density describes the degree of congestion, which means if the value of the probability density is high, there are more hyper-solutions within this area; if the value is low, there are fewer hyper-solutions around. Sparseness of a hyper-solution is defined with the probability density.

Remark 2. *Unlike the other algorithms, we change the parameter h dynamically, which means in the initial phase, this parameter is set to a large value and it decreases with the progressive of time. Just like we point it out above, the probability density curve is smooth when the value of h is large, and the smooth curve means the difference of density value between different hyper-solutions is quite small, so in the next sampling step, every hyper-solution has an almost-same chance to be selected. Later, when h is set to a small value, and a small part of individuals will have a bigger probability to be chosen, and that is to say, the space for sampling is in the constant change. This is one of the reasons we said that we want to keep the exploration-exploitation tradeoff from the perspectives of spatial and time.*

Definition 5. *The sparseness of the hyper-solution p_k is defined by the reciprocal of the probability density:*

$$Sparseness\,(p_k) = \frac{1}{Density\,(p_k)} \tag{3}$$

We would like to make a few remarks here. EDAs generally believe that the Pareto-optimal set follows a certain type of distribution, and some researchers believe that the generating-filter strategy can find this distribution out more effectively than the building-sampling strategy [15]. At the same time, the exploration-exploitation tradeoff strategy has been proven to be an element of solving multi-objective optimization problems, and our idea is to generate more offspring in those areas with lower probability density values to implement the strategy.

More importantly, Importance Sampling is one of the most fundamental techniques for the Monte-Carlo algorithms, and it

[1] Non-dominated set is also called Pareto Set (PS), and the non-dominated front is denoted by PF in this paper.

will help us to overcome a shortcoming caused by the transfer learning technique - the diminishing diversity in the solutions.

We assign each hyper-solution an importance weight, sparseness, to control the probability density of the offspring it can produce. Those hyper-solutions with higher weights have higher probabilities of producing more offspring. It is assumed that each hyper-solution is randomly selected, i.e., x_i follows a uniform distribution $U(x_i) = 1/n$, so let

$$s(x_i) = \mathbf{Pr}(x_i \text{ is selected}) = \frac{sparseness(x_i)}{\sum_k sparseness(x_k)} \quad (4)$$

be the importance distribution. The probability of the number of the offspring produced by the hyper-solution x_i is proportional to $s(x_i)$. According to this analysis, an importance sampling approach based on nonparametric density estimation is designed, and the Procedure NEIS details the approach.

Procedure NEIS(P)

Input: P: a set of hyper-solutions;
Output: a hyper-solution;
for *each individual* p_i *in* P **do**
  Calculate $S\,parseness(p_i)$ by Formula (3);
end
Construct the probabilistic distribution s by Formula (4);
Sample an individual p_k from distribution s;
return p_k

3.3. Probabilistic Inertial Prediction Model

After choosing a hyper-solution, we need an effective means to produce more good offspring. A probabilistic inertial prediction model is presented in this section to achieve the goal. The input of this model is the individual selected by the method described in the above subsection and the output of this model is a set of hyper-solutions. Based on this model, a multi-objective optimization algorithm is proposed.

It consists of two parts to produce an offspring: (i) the distance between the offspring and its parents and (ii) the direction of the hyper-solution. In our method, how to determine the distance is relatively simple, a value λ is sampled from a normal distribution directly, i.e., $\lambda \sim N(0, h \cdot r^n)$, where h is the initial bandwidth in Equation (2), r is the attenuation coefficient and n is the number of iterations. Please note that this distance is gradually decreased. We do this because we believe that it echoes the concept of the exploration-exploitation tradeoff. In the beginning, the searching area is large, and then the search quality can be improved gradually over time.

For the problem of how to decide the direction of the offspring hyper-solution, a special distribution is proposed, called Probabilistic Inertial Distribution (PID), to solve it. Before describing our method in detail, the definition of the probability inertia distribution is given at first.

Definition 6 (Probabilistic Inertial Distribution, PID)**.** *Given a vector $u \in \mathbb{R}^n$, the probabilistic inertial distribution measures the probability of the angle θ between the sampled vector v and u :*

$$\mathrm{PID}_a^{u,v}(\theta) = \frac{e^{-a\theta}}{\int_0^{\pi} e^{-a\theta} d\theta},\ \theta \in [0, \pi], \quad (5)$$

where $a \in \mathbb{R}^+$. u is called the inertial vector.

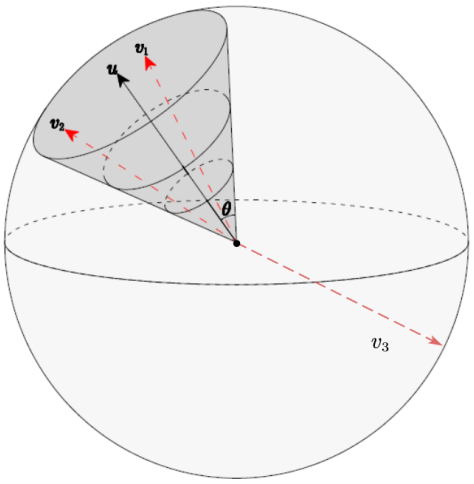

Figure 3: Example of Sampling in Inertial Distribution. The angle between u and v_1 is smaller than the agnle between v_2 and u. The angle between u and u_3 is the largest.

Remark 3. *Please note that there is an infinite number of vectors whose angle to u is θ (the cone in Fig. 3), and we will present an algorithm to select a vector v from them. The function of parameter a is to control the steepness of the curve.*

The purpose of PID is to sample a new vector v according to u. By definition, the probability will decrease when θ increases. Notice that if $a \to 0$, the PID distribution converges to a uniform distribution, that is:

$$\lim_{a \to 0} \mathrm{PID}_a^{u,v}(\theta) = \mathrm{Uniform}([0,\ \pi]). \quad (6)$$

Example 1. *As shown in Fig. 3, u is the inertial vector and three new vectors v_1, v_2 and v_3 are sampled from a given PID distribution according to u. The probability that v_1 is selected is higher than that of v_2 since the angle between u and v_1 is smaller than the agnle between v_2 and u. Obviously, the probability that v_3 is selected is the smallest among the three vectors.*

Definition 7 (Standard Inertial Distribution)**.** *The distribution $\lim_{a \to 0} \mathrm{PID}_a^{u,v}(\theta)$ is called the standard inertial distribution P_0.*

The process of generating offspring is searching for optimal solutions, and this process has time-varying characteristics. In other words, in the beginning of the search, we often want to seek solutions over a large area. As time passed by, we need to narrow down our search for getting high-quality solutions. This is known as the exploration-exploitation tradeoff. For achieving the exploration-exploitation balance, we propose a concept called inertial distribution's amplitude.

Definition 8 (Amplitude of Inertial Distribution)**.** *The amplitude of an inertial distribution P is the* KL *divergence between P and* P_0*:*

$$\mathrm{AMP}(P) = \mathrm{KL}(P, P_0) = \int_0^{\pi} P(\theta) \log \frac{P(\theta)}{P_0(\theta)} d\theta. \quad (7)$$

The amplitude reflects the steepness of the corresponding inertial distribution's curve. When we perform sampling from a distribution with a large amplitude, the vectors which are closer to the inertial vector will be sampled with a higher probability. In contrast, when we carry out sampling from a distribution with a very small amplitude, it is equivalent to sample from a uniform distribution.

The amplitude can be controlled by the parameter a in the Formula (5). So the process of generating offspring can be separated into two phases: start-up phase and refinement phase. In the start-up phase, the amplitude is large, so the efficiency of the algorithm is high. In the refinement phase, the algorithm searches for a better solution in a smaller area since the amplitude will gradually become smaller.

The Procedure GenOffs to be detailed later describes how to generate offspring. The parameter N is a number of offspring the proposed algorithm wants to generate. The value of the number depends on the computing resource available and the problem at hand. The larger the value, the more resources the algorithm requires, and the higher the precision of the solution. After an angle θ is sampled from the inertial distribution, we will carry out the Procedure Randpick to pick out a vector from an infinite number of vectors whose angle to u is θ. And then a step length is sampled from a normal distribution. The Procedure PIPM depicts a multi-objective optimaztion algorithm which is based on the Procedure GenOffs.

3.4. Transfer Learning based Initial Population Generator

The dynamic multi-objective optimization problem is a computation-intensive task which implies it requires a lot of computational resources to search for the POS at a given time. If the knowledge about the POS can be reused to predict future POFs or POSs under different environments, this usually assures performance improvement as well as less computational resource consumption. As a result, we believe that the prediction-based dynamic multi-objective optimization algorithm presents a promising solution.

However, the existing prediction-based algorithms are built on the basis of Independent Identical Distribution (IID) hypothesis, which means that the solutions of a dynamic optimization problems obey an identical distribution. However, it is obvious that the POSs under different environments could follow the different distributions, and this also means that those dynamic optimization algorithms based on the traditional machine learning approach leave much room for improvement. Along this line of thinking, the use of the transfer learning method to develop a novel dynamic estimation of distribution algorithm is proposed.

Pan *et al*. proposed a dimension reduction-based method [30] called Transfer Component Analysis (TCA), to find a low-dimensional space to reduce the difference between source and

Procedure Randpick(u, θ)

Input: u: the inertial vector; θ: the angle between u and the sampled vector;
Output: The sampled vector;

$D_u \leftarrow$ the dimensionality of u;
$First\_nonzero \leftarrow$ the first index of nonzero element in u;
// Construct a linearly independent vector group containing u
for $i = 1$ *to* $D_u - 1$ **do**
  if $i = First\_nonzero$ **then**
    $\beta_i \leftarrow$ a vector with only its D_u-th entry being 1;
  else
    $\beta_i \leftarrow$ a vector with only its i-th entry being 1;
    // Get orthogonal basis R
    Apply Gram-Schmidt process over the martix $[u, \beta_1, \ldots, \beta_{D_u-1}]$ to obtain the orthogonal basis R;
    // Sample from spherical polar coordinates
    Sample a vector v from $D_u - 1$-dimensional spherical polar coordinates;
    // Transform the sample back to original coordinate space
  end
end
return $R^{-1} \cdot [\cos(\theta) v]^T$;

Procedure GenOffs(POS, a, N)

Input: POS: A set of hyper-solutions; a: A control parameter; N: The number of offspring;
Output: A set of hyper-solutions;

$samples \leftarrow \emptyset$;
for $i = 1$ *to* N **do**
  $p \leftarrow \mathrm{NEIS}(POS)$;
  Sample θ from the distribution PID_a;
  $v = \mathrm{Randpick}(p.direction, \theta)$;
  Sample a length λ from $Norm(0, 1/a)$ and normalized it to $[0, \infty]$;
  Add a new hyper-solution $p + \lambda v$ to $samples$;
end
return $samples$;

Procedure PIPM($F_t(X)$, Pop, h, r, N)

Input: $F_t(X)$: a DMOP at time t; Pop: the initialized population; h: the initialized bandwidth; r: the initialized attenuation coefficient; N the number of samplings;

Output: $TempPOS$: the POS of $F_t(X)$;

Get POS^0 and POF^0 by Pop;
$samples$ = GenOffs($Pop, 1/h, N$);
Get POS^1 and POF^1 by $samples$;
$POF = POF^0$;
$POF' = POF^1$;
$TempPOS = POS^1$;
while $\Delta(POF, POF') > \varepsilon$ **do**
    /* The definition of Δ please refer to Equation (8). */
    $samples$ = GenOffs($TempPOS, 1/h, N$);
    $POF = POF'$;
    Get $TempPOS$ and POF' by $samples$;
    $h = h * r$;
end
return $TempPOS$;

targets distributions as well as to preserve the main statistical properties, maximization of data variance. Procedure TCA is a sketch of the TCA algorithm, and interested readers can refer to [30].

Procedure TCA(X, Y, κ)

Input: X, Y: the traing samples of the source and target domains; $\kappa(\cdot,\cdot)$: the kernel function;

Output: W: a transfomation matrix;

Construct the Kernel Matrix $\hat{K}$, the coefficient Matrix L, and the centering Matrix H ;
Construct the Matrix W by using the d leading eigenvectors of $(KLK + \mu I)^{-1}KHK$;
return the matrix W;

In the following Tr-IPG procedure, $F_t(\cdot)$ is the current dynamic optimization function assuming its POS has already been found. F_{t+1} is the optimization problem we want to solve at the next time instance. The major part of the algorithm, Tr-IPG, utilizes the POF at the time t and the transfer learning method to generate a population which can be used to facilitate in searching for the POS at the time $t + 1$.

More specifically, the obtained Pareto-optimal Front (POF) at the time t is regarded as a source domain; the probable solutions of the next time, the time $t + 1$, as the target domain, and then construct a mapping function φ by using the domain adaptation approach. This mapping function will embed the distributions that the source and target domains obey separately into a latent space, and in that space the difference in distributions will become as small as possible. From this, the POS already found can be used to generate an initial population which can be used to search for the POS of the next instance.

Procedure Tr-IPG(F_{t+1}, POF_t, κ)

Input: The Dynamic Optimization Function $F_{t+1}(\cdot)$; $POF_t = \{p_1, \dots p_m\}$ and it is the POF of the function $F_t(\cdot)$ at the time t; a kernel function $\kappa(\cdot,\cdot)$;

Output: A population **Pop-init;**

Initialization;
For the optimization functions $F_t(\cdot)$ and $F_{t+1}(\cdot)$, randomly generate two sets of the solutions X_s and Y_t ; `// Remark 4`
Calculate the objective values of the optimization functions $F_t(X_s)$ and $F_{t+1}(Y_t)$;
W $\leftarrow$ TCA($\{F_t(X_s)\}$, $\{F_{t+1}(Y_t)\}$, κ);
$PLS \leftarrow \emptyset$; `// Remark 5`
for *every* $p \in POF_t$ **do**
    $\kappa_t \leftarrow [\kappa(F_t(p_1), F_t(p)), \cdots, \kappa(F_t(p_m), F_t(p))]^T$;
    $\varphi(F_t(p)) \leftarrow W^T\kappa_t$;
    $PLS = PLS \cup \{\varphi(F_t(p))\}$;
end
for *every* $l \in PLS$ **do**
    $x \leftarrow \underset{x}{\operatorname{argmin}} \|\varphi(F_{t+1}(x)) - l\|$ `// Remark6`
    Pop-init = Pop-init $\cup \{x\}$;
end
return Pop-init;

Remark 4. *The numbers of the samples in X_s and Y_t are predefined. In general, more sampling often means a better result, but it also needs to pay a higher computational cost, so the decision about how many samples needed to be produced in this step depends on the resource available.*

Remark 5. *PLS (in short for POF in the Latent Space) is a set of the mapped solutions in the latent space.*

Remark 6. *We want to find a decision variable x, such that in the latent space, $\varphi(F_{t+1}(x))$ is closest to $l \in PLS$ in the latent space. This also means that we need to solve a single objective optimization problem here, and any single objective optimization algorithm can be applied to solve the problem. In this research, the Interior Point Algorithm is used to solve the problem.*

What the Tr-IPG algorithm outputs is a population, so it can be used as an initial population to search for the POS of the next instance. Please note that we setup a parameter, called the bandwidth h, in our algorithm to control the variances of sampling distributions and the amplitude of PID, and we have discussed the details of the parameter h below Equation (2). The whole process of DANE-EDA is shown in Algorithm 1.

4. The Property Analysis for The DANE-EDA Algorithm

In this section, convergence and computational complexity of the proposed algorithm are analyzed.

Algorithm 1: DANE-EDA: Domain Adaptation and Nonparametric Estimation based EDA

Input: The Dynamic Multi-objective Optimaztion Function $F(X)$; N: the number of samplings; a kernel function $\kappa(\cdot,\cdot)$;

Output: s: the POSs of $F(X)$ at different times;

Randomly initiate the N individuals as the Pop_0;
Initiate bandwidth h;
Initiate attenuation coefficient r;
POS_0 = PIPM($F_0(X)$, Pop_0, r, h, N);
Get POF_0 by POS_0;
$s = POS_0$;
for $t = 1$ *to* n **do**
    Initiate attenuation coefficient r;
    Next-Pop = Tr-IPG($F_t(X)$, POF_{t-1}, $\kappa(\cdot,\cdot)$) ;
    POS_t = PIPM($F_t(X)$,Next-Pop, r, h, N);
    Get POF_t by POS_t;
    $s = POS\, s \cup POS_t$;
end
return s;

4.1. Convergence Analysis

Let's start with the stop condition. We define the following measurement to evaluate the distance of two point sets P_1 and P_2 in the objective space:

$$\Delta(P_1, P_2) = \frac{\sum\limits_{p\in P_1}\min\limits_{q\in P_2}\|p-q\|_2 + \sum\limits_{p\in P_2}\min\limits_{q\in P_1}\|p-q\|_2}{|P_1|+|P_2|}. \quad (8)$$

$\Delta(P_1, P_2)$ can be used to depict the closeness of two Pareto fronts. If P_1 and P_2 are identical, then $\Delta(P_1, P_2) = 0$.

Next, the convergence of DANE-EDA algorithm will be proved . More precisely, we will prove that a sequence of the Pareto Fronts (PFs) computed by the DANE-EDA algorithm is *almost sure convergence* to a value.

Definition 9. *We say that a sequence of X_n converges almost surely toward X means that*

$$\Pr\left(\lim_{n\to\infty} X_n = X\right) = 1.$$

It is usually denoted by

$$\lim_{n\to\infty} X_n \xrightarrow{a.s.} X.$$

Theorem 1. *Let POF_n be the Pareto front got by the DANE-EDA algorithm at the n-th iteration, then*

$$\lim_{n\to\infty} \Delta(POF_n, POF_{n+1}) \xrightarrow{a.s.} 0.$$

Proof. For any hyper-solution p generated by the DANE-EDA at the n-th iteration, the distance between p and its offspring p' satisfies:

$$\begin{aligned}&\lim_{n\to\infty}\left|p'-p\right|\\ &=\lim_{h\to 0}|\lambda_n u_n|\\ &=\lim_{h\to 0}|\lambda_n|,\end{aligned}$$

where u_n is the n-iteration's unit direction vector and λ_n is the corresponding step length.

Since $\lambda_n \sim N(0, h_0 r^n)$, where h_0 is the initial bandwidth (h is defined in Equation 2) and r is the attenuation coefficient, we have

$$\mathrm{E}|\lambda_n|^2 = \mathrm{D}\lambda_n = h_0 r^n,$$

where D is variance.

Using Markov's inequality, the following bound can be obtained , for any given positive real number ε,

$$\Pr\{|\lambda| > \varepsilon\} \le \frac{\mathrm{E}|\lambda|^2}{\varepsilon^2}.$$

So

$$\begin{aligned}&\sum_{n=1}^{\infty}\Pr\{|\lambda_n| > \varepsilon\}\\ &\le \sum_{n=1}^{\infty}\frac{\mathrm{E}|\lambda|^2}{\varepsilon^2}\\ &= \sum_{n=1}^{\infty}\frac{h_0}{\varepsilon^2}r^n,\ \ r\in(0,1)\\ &= \frac{h_0}{\varepsilon^2(1-r)}\\ &< \infty.\end{aligned}$$

By the Borel-Cantelli Lemma [22],

$$\lim_{n\to\infty}|p'-p| \xrightarrow{\text{a.s.}} 0. \quad (9)$$

Let $d(p,q)$ be a distance[2] between p and q in the objective space. Now, according to Equation (9), for every $p \in POF_n$, we have:

$$0 \le \lim_{n\to\infty}\min_{q\in POF_{n+1}} d(p,q) \le \lim_{n\to\infty} d(p,p') \xrightarrow{\text{a.s.}} 0.$$

similarly, for every $q \in POF_{n+1}$, we have:

$$\lim_{n\to\infty}\min_{p\in POF_n} d(p,q) \xrightarrow{\text{a.s.}} 0.$$

According to Equation (8), we obtain that:

$$\lim_{n\to\infty}\Delta(POF_n, POF_{n+1}) \xrightarrow{\text{a.s.}} 0. \quad (10)$$

□

4.2. Computational Time Analysis

At first, let us analyze the static part of the proposed algorithm. Suppose there are m objectives, and evaluating each objective costs $O(1)$ time, then evaluating multiple objectives cost $O(m)$ time. In the proposed algorithm, in every iteration, we sample at most n individuals. For each individual, $O(n)$ time is needed to sample it, so the complexity of sampling at each

[2] In this paper, the Euclidean distance is used as $d(p,q)$

iteration is $O(n^2)$. If P steps are needed to reach the stopping criteria, then the complexity of the static part of the proposed algorithm is $O(mn^2P)$. For the dynamic part of the algorithm, the main computational time is spent on eigenvalue decomposition. It takes $O(d(n_1 + n_2)^2)$ time when d nonzero eigenvectors are to be extracted, where n_1 and n_2 are the numbers of the solutions which are generated to construct the latent space. Overall, DANE-EDA takes $O(n) + O(mn^2P) + O(d(n_1 + n_2)^2$ time to solve an m-objective dynamic optimization problem for every time change.

5. Comparison Studies

5.1. Comparison algorithms

In this research, we compare the DANE-EDA with some chosen state-of-the-art estimation of distribution algorithms. The EGNA algorithm was proposed in [23], and this EDA is based on structure learning and simulation of Gaussian networks. The MBN-EDA [20] is a multi-objective estimation of distribution algorithm and uses the multi-dimensional Bayesian network as its probabilistic model to capture the dependencies between decision variables and target variables. RM-MEDA [45] is a regularity model-based multi-objective estimation of distribution algorithm which assumes that the Pareto set is a piece-wise continuous (m-1)-dimensional (m is the number of objectives) manifold and applies local principal component analysis algorithm to model such a model.

We also compare the proposed algorithm with other dynamic multi-objective optimization algorithms. A random reinitialization method (RND) [29] is implemented as a baseline. MOEA\D-KF was presented in [29], Dynamic NSGA-II (DNSGA-II) [12], the Population Prediction Strategy (PPS) [47] and Multi-objective Optimization algorithm based on Particle Swarm Optimization (MOPSO) [9].

5.2. Performance Metric

We use five performance metrics, variants of Inverted Generation Distance (IGD), a modified Inverted Generation Distance (IGD+), Coverage Scope (CS), Reactivity measure (React), and Hypervolume (Hv), tailored specifically for measuring performance of Dynamic Multi-objective Evolutionary Algorithms. These performance metrics have been adopted widely to quantify the performance of Dynamic Multi-objective Evolutionary algorithms.

- The Inverted Generational Distance (IGD) [38] is used to measure the performance of multi-objective optimization algorithms. Let P^* be the uniformly distributed true Pareto-optimal front and P be the Pareto front acquired by the algorithm. The IGD value is defined as follows:

$$\mathrm{IGD}(P^*, P, C) = \frac{\sum_{v^* \in P^*} \min_{v \in P} \|v^* - v\|}{|P^*|}. \qquad (11)$$

  A smaller IGD value implies a better performance. Please note that the definition of the IGD is slightly different from the original one, and the major difference is the parameter C in Equation (11). The parameter C is a combination of the benchmark functions parameters. We call it as configuration of the benchmark functions. The configurations in our experiments are described in Table 1.

  The MIDG, in short for the Mean value of IGDs, is used to evaluate dynamic multi-objective optimization algorithms, and it takes the average of the IGD values in some time steps over a run as the performance metric, given by

$$\mathrm{MIGD}(P^*, P, C) = \frac{1}{|T|} \sum_{t \in T} \mathrm{IGD}(P^{*t}, P^t, C), \qquad (12)$$

  where P^{*t} and P^t represent the points set of the real POF and the POF obtained by the underlying algorithm at the time t.

  Considering the changing environments a dynamic multi-objective evolutionary algorithm has to perform in, DMIGD, is defined based on MIGD over all considered changing configurations:

$$\mathrm{DMIGD}(P^*, P, C) = \frac{1}{|E|} \sum_{C \in E} \mathrm{MIGD}(P^{*t}, P^t, C), \qquad (13)$$

  where $|E|$ is the number of the different environments and in this research the $|E|$ equals eight. What we want to point out is that the DMIGD can evaluate a dynamic optimization algorithm from a high-level view and it has a significant difference with the MIGD since the MIGD just consider the dynamics in one environment.

- Inverted Generational Distance Plus (IGD+) [17]. Since IGD is not Pareto compliant, it is possible that misleading Pareto incompliant results are obtained. IGD+ considers the Pareto dominance relation between a solution and a reference point when their distance is calculated.

$$d_{IGD+}(v^*, v) = \sqrt{\sum_{k=1}^{m} \left(\max\left\{v_k^* - v_k, 0\right\}\right)^2}, \qquad (14)$$

  where m is the number of the objects. If the obtained solution v is dominated by the point v^* in the true Pareto-optimal front, $d_{IGD+}(v^*, v)$ is equal to the Euclidean distance between v^* and v. Based on this, the definition of IGD+ is given below:

$$\mathrm{IGD+}(P^*, P, C) = \frac{\sum_{v^* \in P^*} \min_{v \in P} d_{IGD+}(v^*, v)}{|P^*|}. \qquad (15)$$

  P^* is the uniformly distributed true Pareto-optimal front and P is the obtained Pareto front through the algorithm.The parameter C is a combination of the benchmark functions parameters. DMIGD+ has a similar meaning to the DMIGD. A smaller DMIGD+ value implies a better performance.

- Coverage Scope (CS) [46] is used to measure the diversity of solutions. The CS value of P is defined by:

$$\mathrm{CS}(P) = \frac{1}{|P|}\sum_{i=1}^{|P|}\max\left\{\left\|f(x_i) - f(x_j)\right\|\right\}, \qquad (16)$$

where P is the obtained optimal Pareto front. x_i, $x_j \in P$ with $i \geq 1$ and $j \leq |P|$. This index describes the diversity of the solution set. A larger CS value indicates a better diversity. Its variant, DMCS, has a similar meaning to the DMIGD.

- Reactivity measure (React) [39] is used to measure the robustness of algorithms. The definition is given below:

$$React(t) = \min\left\{t' - t | t < t' \in \mathbb{N}, \frac{acc(t')}{acc(t)} \geq 1 - \varepsilon\right\}, \quad (17)$$

where $acc(t) = \frac{\mathrm{Hv}(POF(t))}{\max \mathrm{Hv}(POF)}$ represents the accuracy of Pareto front at time t, and Hv means Hypervolume [49]. This index describes the needed time to recover the Pareto front. Smaller React value indicates better robustness. Its variant, DMReact, has a similar meaning to the DMIGD.

- Hypervolume [49] has been gaining a lot of interest in recent years and it is a set measure reflecting the volume enclosed by a Pareto front approximation. Let P be a non-dominated set and the hypervolume indicator for P, denoted as Hv(P), is dependent upon the reference point $r = (r_1, r_2, ..., r_k)$ and is formalized as

$$\mathrm{Hv}\,(P) = Leb\left(\bigcup_{x\in P}\left[f_1\,(x)\,, r_1\right]\times\cdots\times\left[f_k\,(x)\,, r_k\right]\right), \quad (18)$$

where $Leb(P)$ is the Lebesgue measure of the set P, and $\left[f_1\,(x)\,, r_1\right]\times\left[f_2\,(x)\,, r_2\right]\times\cdots\times\left[f_k\,(x)\,, r_k\right]$ is the hypercube containing all criterion vectors that are weakly dominated by the elements $x \in P$ and themselves dominate the reference point r. The non-dominated set which has a relatively uniform and widely distribution would have a larger corresponding Hypervolume value. DMHv has a similar meaning to the DMIGD.

5.3. Test Instances and Experimental Settings

The IEEE CEC 2015 benchmark problems set was chosen as test functions and the problem set has twelve testing functions. The POFs of the testing functions have different shapes and each function belongs to a certain DMOPs type. Fig. 4 describes the true POFs of the twelve testing functions and we let the functions change three times.

In the definitions, the decision variables are $x = (x_1, \ldots, x_n)$ and $t = \frac{1}{n_t}\left\lfloor\frac{\tau_T}{\tau_t}\right\rfloor$, where n_t, τ_T, and τ_t are the severity of change, maximum number of iterations, and frequency of change respectively. Table 1 describes the different combinations of n_t, τ_t, and τ_T used in our experiments. Please note that, for each n_t-τ_T combination, there will be $\frac{\tau_T}{\tau_t}$ environment changes. In other words, in all of our experiments, there are altogether five changes for the twelve dynamic problems.

Table 1: Environment Settings

n_t	τ_t	τ_T
10	5	25
10	10	50
10	25	125
10	50	250
1	5	25
1	10	50
20	25	125
20	50	250

The POFs of the testing functions have different shapes and each function belongs to a certain DMOPs type. For example, the POF of the functions could be non-convex, convex, isolated, deceptive, continuous or discontinuous. Table 2 describes the types of the testing functions. Type I implies POS changes, but POF does not change; Type II means that the POS and the POF change as well; Type III means that the POF changes, but the POS does not change.

FDA4 and FDA5 are 3-objective functions while all of the remaining are 2-objective functions. FDA5$_{iso}$ is a Type II DMOP and its POF is isolated and non-convex. Spread of solutions in the POF of this function changes over time. FDA5$_{dec}$ is also a Type II DMOP but has a deceptive and non-convex POF. DIMP2 is a Type I DMOP and it has a convex POF. dMOP2 is a Type II DMOP and its POF changes from convex to concave, and vice versa. dMOP2$_{iso}$ is also a Type II DMOP and has an isolated POF. dMOP2$_{dec}$ has a deceptive POF and its POF changes from convex to concave and vice versa. dMOP3 is a Type I DMOP. It has a convex POF and spread of the POF changes over time. HE7 and HE9 are type III DMOPs and their POS are not dependent on time. HE2 is also a type III DMOP, and it has discontinuous POF, with various disconnected continuous sub-regions.

The dimensions of the decision variables are from 10 to 30. Please note that the A, B and C values for the functions FDA5$_{iso}$, FDA5$_{dec}$, dMOP2$_{iso}$ and dMOP2$_{dec}$ are set to G(t), 0.001 and 0.05 respectively.

In all of the experiments, the population size is set to 200 and in each generation every algorithm will generate no more than 200 solutions. For the TCA parameter, we set the Gaussian kernel function as the default and the expected dimensionality was set to be 20. The value of μ was set to 0.5. The specific parameter settings follow instructions in the original papers. In EGNA [23], truncation selection is chosen as the selection method. MBN-EDA[20] uses profit gain ordering as ranking method. In RM-MEDA[45], the number of clusters in local PCA algorithm is set to be 5.

Please note that our experimental results are obtained without explicitly tuning the TCA algorithm's parameters one by one, and if we adjust the parameters specifically for different test functions, we have reason to believe that we can obtain better experimental results. The reason that we did not tune the parameters separately for getting better results is that the twelve test functions are not exactly the same, so we can set different

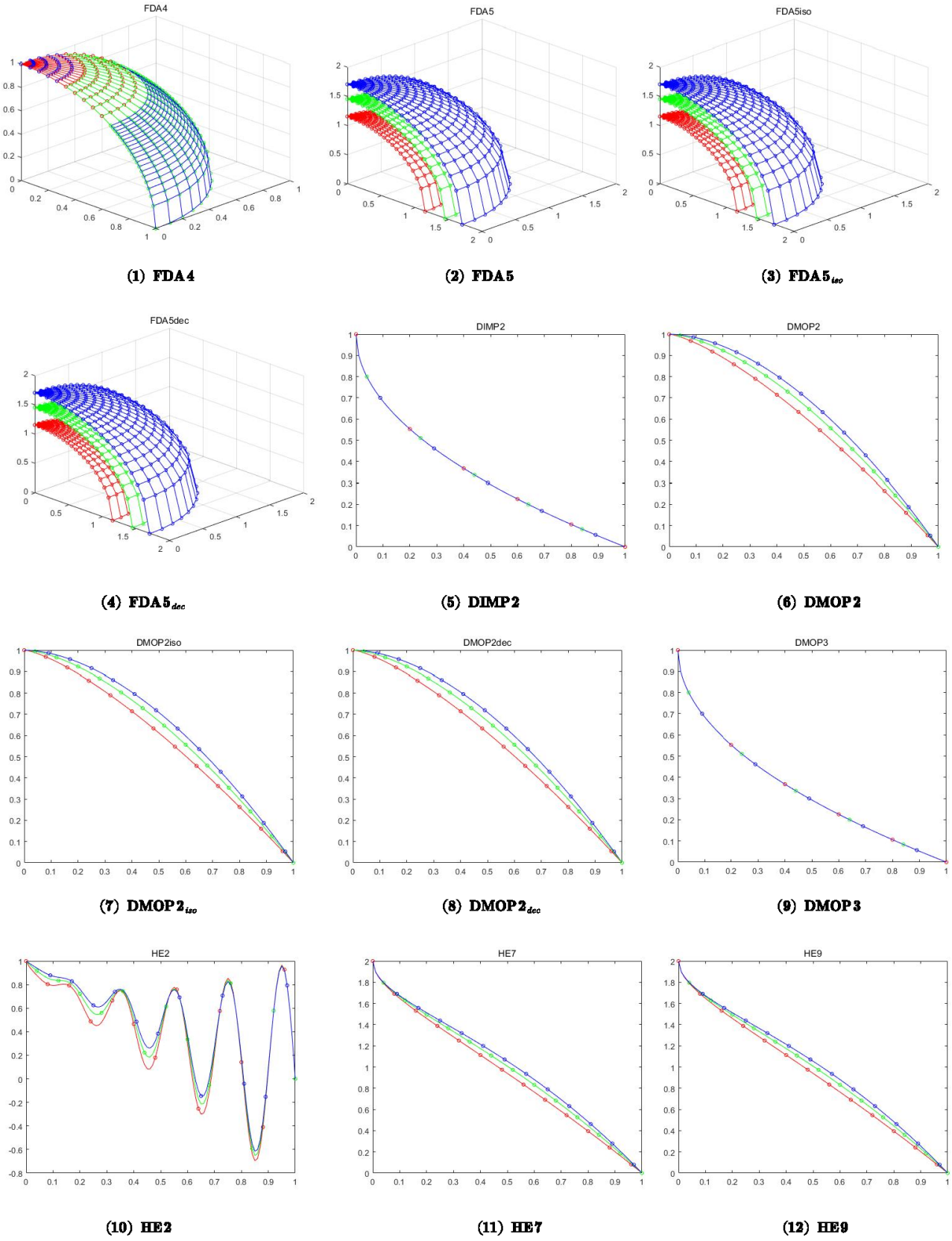

Figure 4: The true POFs of the twelve testing functions

Table 2: Characteristics of the test functions

Name	Decision Variable Dimension	Objectives	Properties of the DMOP
FDA4	12	3	TYPE I, Non-convex POF
FDA5	12	3	TYPE II, Non-convex POF
$FDA5_{iso}$	12	3	TYPE II, Isolated, non-convex POF
$FDA5_{dec}$	12	3	TYPE II, Deceptive, non-convex POF
DIMP2	10	2	TYPE I, Convex POF
dMOP2	10	2	TYPE II, POF changes from convex to concave
$dMOP2_{iso}$	10	2	TYPE II, Isolated POF, POF changes from convex to concave
$dMOP2_{dec}$	10	2	TYPE II, Deceptive POF, POF changes from convex to concave
dMOP3	10	2	TYPE I, Convex POF, Spread of the P OF solutions changes over time
HE2	30	2	TYPE III, Discontinuous POF, with various disconnected continuous sub-regions
HE7	10	2	TYPE III, POF changes from convex to concave
HE9	10	2	TYPE III,POF changes from convex to concave

parameters to get the best performance for each test function. For example, we can construct different latent spaces for each of the twelve benchmark functions individually via setting different parameters of the TCA method. However, we think that this one-by-one-adjustment strategy does not effectively explain the versatility of the proposed design.

6. Results and Discussion

6.1. Comparison Results

For each benchmark functions, eight different n_t-τ_t combinations listed in Table 1 are tested and the different performance metrics are calculated. Owing to space limitation, we just report DMIGD, DMCS, DMReact and DMHv values in Table 3 to 7. These values are obtained by averaging different changes and then averaging over different configurations.

Table 3 records the DMIGD values of the algorithms running on different testing functions under different configurations. For the twelve test functions, the DANE-EDA achieves the best results in $FDA5_{iso}$, $FDA5_{dec}$, DIMP2, HE7 and HE9, at the same time, for the other four benchmark functions, FDA4, FDA5, $dMop2_{iso}$ and HE2, the performance of the proposed algorithm is very near that of the best algorithm. The experiments show that the proposed algorithm is competitive at handling the Types II and III DMOPs or the problems that its POF changes over time since these test functions belong in these types.

We think that the reason why DANE-EDA has advantages in dealing with POF-changing optimization problems is largely due to the transfer learning-based initial population generation method. In other words, our method establishes a link between the POF at the time t and the POF at the time $t + 1$, and the experiments prove that our idea, the transfer learning plus estimation of distribution algorithm, can effectively improve the performance of handling dynamic multi-objective optimization problems.

The performance quantified according to DMCS is given in Table 5. In four cases, dMOP2, $dMOP2_{dec}$, HE7 and HE9, the performance of the DANE-EDA is the best among all competing algorithms, and MOPSO and MOEA\D-KF each win the three other test functions.

According to the definitions, we know that dMOP2 is a type II DMOP and its time-varying POS is sinusoidal in nature. $dMOP2_{iso}$ and $dMOP2_{dec}$ consist of isolated and deceptive POF respectively, and we have mentioned HE2, HE7 and HE9 above. By observing the common characteristics of these functions, we believe that the reason why DANE-EDA has better performance over these functions according to DMCS metric is that the nonparametric density estimation based importance sampling method has played a role, especially the integration of the sampling method and the transfer learning is particularly applicable to the Type II or Type III dynamic multi-objective optimization problems.

In addition, as shown in Table 5, we can find that DANE-EDA is not tailored specifically for DMOPs with isolated POF. This may be because the isolated POFs are usually flat and DANE-EDA cannot move fast and cover the whole POF since the bandwidth parameter is decreasing over time.

Table 6 gives out the DMReact performance for each method. The value is also averaged over time and changes. As we described in its definition, the value depicts the time needed for an algorithm to recover the Pareto front as environment has changed, and it is an index for robustness of an algorithm. The DANE-EDA achieves the best results in FDA4, HE7 and HE9, while comes near the top performers in FDA5, $dMOP2_{dec}$ and HE2 testing functions.

The statistical values of DMHv for DANE-EDA in comparison with other nine algorithms are presented in Table 7. These data indicate that the DANE-EDA performs the best in five benchmark functions, and at the same time, for the HE2, dMOP3, and $FDA5_{iso}$ functions, there is little difference in performance between the proposed algorithm and the best ones. This result indicates that the diversity of solutions can be improved by combining importance sampling with transfer learning. We can see that in almost all cases the DANE-EDA algorithm have achieved relatively good results.

Table 3: Statistical results of the proposed algorithm, DANE-EDA, and nine chosen competing algorithms in term of DMIGD performance metric

DMIGD	**DANE-EDA**	**NSGA-II-A**	**NSGA-II-B**	**MOPSO**	**RMMEDA**	**MBNEDA**	**EGNA**	**PPS**	**RND**	**MOEA\D-KF**
FDA4	0.075	0.311	0.279	0.082	**0.069**	0.379	0.441	0.070	0.178	0.181
FDA5	0.192	0.372	0.371	0.152	0.126	0.692	0.632	**0.125**	0.265	0.306
$FDA5_{iso}$	**0.052**	0.084	0.088	0.077	0.054	0.689	0.304	0.054	0.103	0.100
$FDA5_{dec}$	**0.145**	0.585	0.543	0.236	0.360	1.446	1.433	0.178	0.305	0.295
DIMP2	**3.158**	3.691	4.304	5.390	4.971	11.818	7.311	4.964	4.505	5.069
dMOP2	4.055	1.221	1.179	**0.061**	3.647	5.702	5.308	3.625	1.044	1.487
$dMOP2_{iso}$	0.024	0.027	0.026	0.027	0.024	0.150	0.027	0.024	0.019	**0.019**
$dMOP2_{dec}$	3.913	0.729	0.782	0.417	**0.195**	12.873	9.115	10.386	0.996	3.477
dMOP3	3.972	0.187	1.089	**0.048**	3.670	5.522	5.111	3.614	5.816	6.798
HE2	0.245	0.212	0.216	0.313	0.833	2.308	1.645	0.667	**0.165**	0.180
HE7	**0.028**	0.082	0.083	0.092	0.033	2.134	0.329	0.164	0.133	0.151
HE9	**0.231**	0.285	0.284	0.235	0.249	1.299	0.334	0.274	0.312	0.332

Table 4: Statistical results of the proposed algorithm, DANE-EDA, and nine chosen competing algorithms in term of DMIGD+ performance metric

DMIGD+	**DANE-EDA**	**NSGA-II-A**	**NSGA-II-B**	**MOPSO**	**RMMEDA**	**MBNEDA**	**EGNA**	**PPS**	**RND**	**MOEA\D-KF**
FDA4	**0.054**	0.246	0.217	0.065	0.055	0.311	0.371	0.055	0.137	0.144
FDA5	0.117	0.240	0.234	**0.064**	0.087	0.479	0.479	0.085	0.179	0.208
$FDA5_{iso}$	0.024	0.036	0.036	0.027	**0.022**	0.347	0.123	**0.022**	0.040	0.040
$FDA5_{dec}$	**0.081**	0.514	0.486	0.162	0.359	1.349	1.317	0.145	0.198	0.209
DIMP2	**3.124**	3.678	4.291	5.371	4.767	11.818	7.293	4.952	4.448	5.216
dMOP2	4.022	1.189	1.158	**0.042**	3.658	5.694	5.267	3.625	1.332	2.102
$dMOP2_{iso}$	**0.001**	**0.001**	**0.001**	0.003	**0.001**	0.077	0.004	**0.001**	0.011	0.011
$dMOP2_{dec}$	3.791	0.698	0.735	0.341	**0.169**	12.873	9.081	10.385	0.948	3.161
dMOP3	3.947	0.152	1.056	**0.024**	3.663	5.520	5.085	3.614	5.706	6.572
HE2	0.230	0.205	0.212	0.298	0.828	2.308	1.643	0.666	**0.142**	0.157
HE7	**0.026**	0.080	0.081	0.088	0.031	1.373	0.244	0.142	0.125	0.151
HE9	**0.226**	0.280	0.279	0.251	0.243	1.158	0.304	0.259	0.299	0.311

Table 5: Statistical results of the proposed algorithm, DANE-EDA, and nine chosen competing algorithms in term of DMCS performance metric

DMCS	DANE-EDA	NSGA-II-A	NSGA-II-B	MOPSO	RM-MEDA	MBNEDA	EGNA	PPS	RND	MOEA\D-KF
FDA4	2.071	2.243	2.265	1.369	1.950	1.738	2.436	2.146	3.275	**3.488**
FDA5	2.985	2.631	2.523	2.066	2.357	1.988	2.810	2.451	3.628	**3.896**
$FDA5_{iso}$	1.716	1.652	1.645	**1.762**	1.705	1.372	1.723	1.706	1.731	1.729
$FDA5_{dec}$	3.163	3.013	2.978	**376.865**	3.933	3.823	5.371	33.975	2.760	2.904
DIMP2	6.246	2.950	2.959	5.660	5.311	6.621	8.490	5.680	16.588	**16.597**
dMOP2	**15.073**	4.015	4.085	7.629	1.530	3.294	2.520	1.442	7.809	7.272
$dMOP2_{iso}$	1.082	1.077	1.076	1.141	1.083	**1.249**	1.029	1.083	1.086	1.086
$dMOP2_{dec}$	**2817.500**	1.238	1.027	1.920	0.945	8.822	8.263	2.554	5.986	5.042
dMOP3	5.485	5.143	4.275	**6.469**	1.493	2.471	2.668	1.435	5.415	5.946
HE2	1.505	**2.061**	1.993	2.002	1.478	0.486	1.497	1.468	1.768	1.654
HE7	**9.024**	8.117	7.826	6.080	6.700	4.509	1.713	6.088	8.621	8.626
HE9	**9.453**	8.300	8.055	7.216	8.100	4.021	4.401	8.134	9.368	9.226

Table 6: Statistical results of the proposed algorithm, DANE-EDA, and nine chosen competing algorithms in term of DMReact performance metric

DMReact	DANE-EDA	NSGA-II-A	NSGA-II-B	MOPSO	RM-MEDA	MBNEDA	EGNA	PPS	RND	MOEA\D-KF
FDA4	**1.094**	1.828	1.641	1.234	1.578	1.734	1.500	1.469	1.563	1.516
FDA5	1.219	1.563	1.531	**1.172**	1.688	1.750	1.297	1.703	1.813	1.797
$FDA5_{iso}$	1.078	1.188	1.172	1.078	**1.000**	1.609	1.203	**1.000**	1.047	1.047
$FDA5_{dec}$	1.375	1.281	1.313	1.578	1.250	1.906	**1.234**	1.625	1.859	1.859
DIMP2	1.453	1.422	1.500	1.344	1.438	**1.219**	1.391	1.656	1.563	1.438
dMOP2	1.406	1.484	1.438	1.266	1.500	**1.188**	1.750	1.453	1.500	1.625
$dMOP2_{iso}$	1.188	1.125	1.141	1.141	1.172	1.188	1.203	1.156	**1.109**	1.156
$dMOP2_{dec}$	1.203	1.516	1.578	1.609	1.313	**1.188**	1.391	1.344	1.484	1.484
dMOP3	1.484	1.344	1.297	1.156	**1.094**	1.750	1.406	1.109	1.297	1.266
HE2	1.250	1.250	1.391	1.250	1.375	1.281	1.344	**1.172**	1.250	1.313
HE7	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	1.250	1.203	**1.000**	**1.000**	**1.000**
HE9	**1.000**	1.016	**1.000**	**1.000**	**1.000**	1.203	1.031	**1.000**	**1.000**	**1.000**

Table 7: Statistical results of the proposed algorithm, DANE-EDA, and nine chosen competing algorithms in term of DMHv performance metric

DMHv	DANE-EDA	NSGA-II-A	NSGA-II-B	MOPSO	RM-MEDA	MBNEDA	EGNA	PPS	RND	MOEA\D-KF
FDA4	0.547	0.639	0.641	0.488	0.769	0.426	0.585	0.760	0.787	**0.799**
FDA5	0.672	0.621	0.610	0.579	0.684	0.396	0.623	0.670	0.711	**0.715**
$FDA5_{iso}$	0.439	0.426	0.424	**0.458**	0.435	0.082	0.298	0.433	0.383	0.385
$FDA5_{dec}$	0.736	0.662	0.674	0.728	**0.830**	0.481	0.712	0.671	0.588	0.593
DIMP2	0.328	0.245	0.236	0.464	0.400	0.009	0.421	0.386	**0.600**	0.595
dMOP2	**0.702**	0.317	0.356	0.498	0.403	0.049	0.416	0.409	0.597	0.581
$dMOP2_{iso}$	**0.426**	0.425	0.424	0.419	0.421	0.243	0.408	0.421	0.414	0.414
$dMOP2_{dec}$	**0.787**	0.285	0.259	0.526	0.358	0.022	0.449	0.395	0.534	0.524
dMOP3	0.692	0.548	0.535	**0.701**	0.642	0.057	0.561	0.643	0.527	0.539
HE2	0.621	0.605	0.596	0.590	0.426	0.010	0.335	0.461	**0.672**	0.662
HE7	**0.904**	0.884	0.881	0.860	0.871	0.173	0.544	0.835	0.871	0.867
HE9	**0.902**	0.885	0.883	0.885	0.889	0.256	0.820	0.885	0.890	0.886

Our views on this issue are as follows. In one respect, probabilistic inertia model can help to obtain more heuristic information from environment and speed up the updating process of searching for Pareto front, leading to better performance (smaller DMIGD value). Meanwhile, the step width and sampling angle of the model will reduce robustness of the algorithm in some sense since it would need some time to adjust the parameters when environment changes.

7. Conclusion

In this article, a novel estimation distribution algorithm, DANE-EDA, is proposed for solving dynamic multi-objective optimization problems. The uniqueness of the proposed algorithm is that it makes an attempt at unifying importance sampling, nonparametric density estimation, probabilistic prediction mechanism and domain adaptation technique under one framework, so that it can take the advantage of Monte-Carlo method and transfer learning technologies.

The merits of this seamless integration are reflected in two aspects. First, the transfer learning based initial population generation method can make the algorithm reuse the knowledge gained, such that the efficiency of the dynamic multi-objective optimization algorithm is improved. Secondly, transfer learning will inadvertently deteriorate the diversity in solutions, and we suspect that this is due to the latent space mapping. However, the non-parametric probability density estimation based importance sampling can overcome this problem as well as preserve the exploration-exploitation tradeoff in some degree.

After presenting a detailed description of the algorithm, the convergence and analysis of the computational complexity of the proposed algorithm are proved. In the experimental part, the IEEE CEC 2015 benchmark problems set is taken as test functions and the problem set has twelve testing functions. The proposed algorithm and the other nine state-of-the-art algorithms have been evaluated by five performance metrics. The experimental results show that the convergence and diversity of the proposed algorithm is promising, but at the same time, we also find that the robustness and computational speed of the algorithm remain a critical issue to be improved. However, we believe this line of research proposed herein inspires further innovations in solving real-world application with various degrees of complexities and uncertainties.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.61673328). The first author also wants to thank China Scholarship Council (No. 20150631505) and Oklahoma State University for providing funding and facilities to support his research as a visiting scholar.

References

[1] Arqub, O. A., Abo-Hammour, Z., 2014. Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm. Information sciences 279, 396–415.

[2] Azzouz, R., Bechikh, S., Said, L. B., 2017. A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy. Soft Computing 21 (4), 885–906.

[3] Azzouz, R., Bechikh, S., Said, L. B., 2017. Dynamic multi-objective optimization using evolutionary algorithms: a survey. In: Recent Advances in Evolutionary Multi-objective Optimization. Springer, pp. 31–70.

[4] Bosman, P., 2007. Learning and anticipation in online dynamic optimization. Evolutionary computation in dynamic and uncertain environments, 129–152.

[5] Branke, J., 1999. Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the 1999 Congress on Evolutionary Computation. Vol. 3. IEEE, pp. 1875–1882.

[6] Branke, J., 2012. Evolutionary optimization in dynamic environments. Vol. 3. Springer Science & Business Media.

[7] Branke, J., Kaußler, T., Smidt, C., Schmeck, H., 2000. A multi-population approach to dynamic optimization problems. In: Evolutionary Design and Manufacture. Springer, pp. 299–307.

[8] Cobb, H. G., 1990. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Tech. Rep. Technical Report AIC-90-001.

[9] Coello, C. C., Lechuga, M. S., 2002. Mopso: A proposal for multiple objective particle swarm optimization. In: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on. Vol. 2. IEEE, pp. 1051–1056.

[10] Costa, M., Minisci, E., 2003. Moped: a multi-objective parzen-based estimation of distribution algorithm for continuous problems. In: Evolutionary Multi-Criterion Optimization. Springer, pp. 282–294.

[11] 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: International conference on evolutionary multi-criterion optimization. Springer, pp. 803–817.

[12] 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: International Conference on Evolutionary Multi-Criterion Optimization. Springer, pp. 803–817.

[13] Goh, C.-K., Tan, K. C., 2009. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Transactions on Evolutionary Computation 13 (1), 103–127.

[14] Grefenstette, J., 1992. Genetic algorithms for changing environments, 137–144.

[15] Hauschild, M., Pelikan, M., 2011. An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation 1 (3), 111–128.

[16] Huang, Z., Jiang, M., 2016. Trend prediction model based multi-objective estimation of distribution algorithm,(in chinese). Artificial Intelligence and Robotics Research 5 (1), 1–12.

[17] Ishibuchi, H., Masuda, H., Nojima, Y., 2015. A study on performance evaluation ability of a modified inverted generational distance indicator. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. ACM, pp. 695–702.

[18] Jiang, M., Huang, Z., Qiu, L., Huang, W., Yen, G. G., 2017. Transfer learning based dynamic multiobjective optimization algorithms. IEEE Transactions on Evolutionary Computation PP (99), 1–1.

[19] Jiang, S., Yang, S., 2017. A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization. IEEE Transactions on Evolutionary Computation 21 (1), 65–82.

[20] Karshenas, H., Santana, R., Bielza, C., Larranaga, P., 2014. Multiobjective estimation of distribution algorithm based on joint modeling of objectives and variables. IEEE Transactions on Evolutionary Computation 18 (4), 519–542.

[21] Khan, N., Goldberg, D. E., Pelikan, M., 2002. Multi-objective bayesian optimization algorithm. Urbana 51, 61801.

[22] Klenke, A., 2013. Probability theory: a comprehensive course. Springer Science & Business Media.

[23] Larrañaga, P., Etxeberria, R., Lozano, J. A., Peña, J. M., 2000. Optimization in continuous domains by learning and simulation of gaussian networks, 201–204.

[24] Larrañaga, P., Karshenas, H., Bielza, C., Santana, R., 2012. A review on probabilistic graphical models in evolutionary computation. Journal of Heuristics 18 (5), 795–819.

[25] Larranaga, P., Lozano, J. A., 2002. Estimation of distribution algorithms: A new tool for evolutionary computation. Vol. 2. Springer Science & Business Media.

[26] Li, C., Yang, S., 2008. Fast multi-swarm optimization for dynamic optimization problems. In: Natural Computation, 2008. ICNC'08. Fourth International Conference on. Vol. 7. IEEE, pp. 624–628.

[27] Liu, R., Li, J., Mu, C., Jiao, L., et al., 2017. A coevolutionary technique based on multi-swarm particle swarm optimization for dynamic multi-objective optimization. European Journal of Operational Research 261 (3), 1028–1051.

[28] Merton, R. C., Samuelson, P. A., 1992. Continuous-time finance.

[29] Muruganantham, A., Tan, K. C., Vadakkepat, P., 2016. Evolutionary dynamic multiobjective optimization via kalman filter prediction. IEEE Transactions on Cybernetics 46 (12), 2862–2873.

[30] Pan, S. J., Tsang, I. W., Kwok, J. T., Yang, Q., 2011. Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks 22 (2), 199–210.

[31] Pan, S. J., Yang, Q., 2010. A survey on transfer learning. IEEE Transactions on knowledge and data engineering 22 (10), 1345–1359.

[32] Pelikan, M., 2005. Bayesian optimization algorithm. In: Hierarchical Bayesian optimization algorithm. Springer, pp. 31–48.

[33] Pelikan, M., Sastry, K., Goldberg, D. E., 2005. Multiobjective hboa, clustering, and scalability. In: Proceedings of the 7th annual conference on Genetic and evolutionary computation. ACM, pp. 663–670.

[34] Raquel, C., Yao, X., 2013. Dynamic multi-objective optimization: a survey of the state-of-the-art. In: Evolutionary Computation for Dynamic Optimization Problems. Springer, pp. 85–106.

[35] Rossi, C., Abderrahim, M., Díaz, J. C., 2008. Tracking moving optima using kalman-based predictions. Evolutionary computation 16 (1), 1–30.

[36] Shah, R., Reed, P., 2011. Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems. European Journal of Operational Research 211 (3), 466–479.

[37] Shawe-Taylor, J., Cristianini, N., 2004. Kernel methods for pattern analysis. Cambridge university press.

[38] Sierra, M. R., Coello, C. A. C., 2005. Improving pso-based multi-objective optimization using crowding, mutation and-dominance. In: Evolutionary Multi-Criterion Optimization. Springer, pp. 505–519.

[39] Sola, M. C., 2010. Parallel processing for dynamic multi-objective optimization. Ph.D. thesis, Ph. D. thesis, Universidad de Granada, Dept. of Computer Architecture and Computer Technology, Universidad de Granada, Spain.

[40] Stroud, P. D., 2001. Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations. IEEE Transactions on Evolutionary Computation 5 (1), 66–77.

[41] Vavak, F., Fogarty, T. C., Jukes, K., 1996. A genetic algorithm with variable range of local search for tracking changing environments, 376–385.

[42] Wang, Y., Li, B., 2010. Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. Memetic Computing 2 (1), 3–24.

[43] Woldesenbet, Y. G., Yen, G. G., 2009. Dynamic evolutionary algorithm with variable relocation. IEEE Transactions on Evolutionary Computation 13 (3), 500–513.

[44] Yang, S., Li, C., 2010. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. IEEE Transactions on Evolutionary Computation 14 (6), 959–974.

[45] Zhang, Q., Zhou, A., Jin, Y., 2008. Rm-meda: A regularity model-based multiobjective estimation of distribution algorithm. IEEE Transactions on Evolutionary Computation 12 (1), 41–63.

[46] Zhang, Z., Qian, S., 2011. Artificial immune system in dynamic environments solving time-varying non-linear constrained multi-objective problems. Soft Computing 15 (7), 1333–1349.

[47] Zhou, A., Jin, Y., Zhang, Q., 2014. A population prediction strategy for evolutionary dynamic multiobjective optimization. IEEE Transactions on Cybernetics 44 (1), 40–53.

[48] Zhou, L., Zhou, A., Zhang, G., Shi, C., 2011. An estimation of distribution algorithm based on nonparametric density estimation. In: 2011 IEEE Congress on Evolutionary Computation (CEC' 2011). IEEE, pp. 1597–1604.

[49] Zitzler, E., Thiele, L., 1998. Multiobjective optimization using evolutionary algorithmsa comparative case study. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 292–301.

[50] Zou, J., Li, Q., Yang, S., Bai, H., Zheng, J., 2017. A prediction strategy based on center points and knee points for evolutionary dynamic multi-objective optimization. Applied Soft Computing 61, 806–818.