Multi-Population Evolution based Dynamic Constrained Multiobjective Optimization under Diverse Changing Environments

Qingda Chen, *Member, IEEE*, Jinliang Ding, *Senior Member, IEEE*, Gary G. Yen, *Fellow, IEEE*, Shengxiang Yang, *Senior Member, IEEE*, and Tianyou Chai, *Life Fellow, IEEE*

***Abstract*—Dynamic constrained multiobjective optimization involves irregular changes in the distribution of the true Pareto-optimal fronts, drastic changes in the feasible region caused by constraints, and the movement directions and magnitudes of the optimal distance variables due to diverse changing environments. To solve these problems, we propose a multi-population evolution based dynamic constrained multiobjective optimization algorithm. In this algorithm, we design a tribe classification operator to divide the population into different tribes according to a feasibility check and the objective values, which is beneficial for driving the population toward the feasible region and Pareto-optimal fronts. Meanwhile, a population selection strategy is proposed to identify promising solutions from tribes and exploit them to update the population. The optimal values of the distance variables vary differently with dynamic environments, thus, we design a dynamic response strategy for solutions in different tribes that estimates their distances to approach the Pareto-optimal fronts and regenerates a promising population when detecting environmental changes. In addition, a scalable generator is designed to simulate diverse movement directions and magnitudes of the optimal distance variables in real-world problems under dynamic environments, obtaining a set of improved test problems. Experimental results show the effectiveness of test problems, and the proposed algorithm is impressively competitive with several chosen state-of-the-art competitors.**

***Index Terms*—Dynamic constrained multiobjective optimization, tribe classification operator, population selection, dynamic response**

I. Introduction

MANY real-world applications, such as fluid catalytic cracking-distillation processes [1], [2] and the manufacture of fused magnesia [3], involve at least two conflicting objectives, constraints as well as time-changing design parameters [4], and they can be called dynamic constrained multiobjective optimization problems (DCMOPs). Their objective values must be optimized simultaneously while satisfying various constraint functions when dynamic environments are detected.

A decision variable vector (i.e., a solution) of a DCMOP is a feasible solution if it satisfies all constraint functions; otherwise, it is an infeasible solution. For two solutions ($\mathbf{x}_1$ and $\mathbf{x}_2$), $\mathbf{x}_1$ is said to dominate $\mathbf{x}_2$ (i.e., $\mathbf{x}_1 \prec \mathbf{x}_2$) if all objective values of the former are not worse than those of the latter, and $f_b(\mathbf{x}_1, t)$ is less than $f_b(\mathbf{x}_2, t)$ for at least one objective index $b \in \{1,\ldots,m\}$. $\mathbf{x}_1$ is a nondominated solution (also called a Pareto-optimal solution) if there is no other solution to dominate it. Unlike single-objective optimization problems, a set of solutions can satisfy the optima of a given DCMOP, and they are called Pareto-optimal solutions. The set of all Pareto-optimal solutions at environment t is referred to as the dynamic Pareto-optimal set (DPOS) at t. The image of the DPOS in the objective space is called the dynamic Pareto-optimal front (DPOF) under environment t. A decision variable in a Pareto-optimal solution is considered to be a distance variable if changing x_i in $\mathbf{x}$ can only result in a decision vector that equals $\mathbf{x}$, dominates $\mathbf{x}$, or is dominated by $\mathbf{x}$ [6]. Instead, if changing x_i in $\mathbf{x}$ can only cause a vector that is incomparable or equivalent to $\mathbf{x}$, then x_i is called a position variable. In other words, changing a position variable on its own never causes a dominated or dominating decision vector, while changing a distance variable on its own will never cause incomparable decision vectors. An example of a DCMOP including position and distance variables is given in Section S-I of the Supplementary Material. This paper focuses on the optimization of the distance variables in DCMOPs.

Diverse changing environments complicate the convexity and continuity of the true DPOFs, the feasible region of the objective values (hereafter referred to simply as the feasible region), and the optimal values of the distance variables. The detailed problem characteristics are as follows.

1) Dynamic environments lead to changes in the concavity and convexity of the unconstrained DPOFs (i.e., the DPOFs of dynamic multiobjective optimization problems (DMOPs) without constraint functions), resulting in the discontinuity of the true DPOFs. The feasible region varies with constraint functions and causes the infeasibility of the unconstrained DPOFs, narrowing and fragmenting the true DPOFs and making it difficult for algorithms to search for

This work was supported in part by the National Natural Science Foundation of China under Grant 62203101, Grant 61988101, Grant 61991400, Grant 61991403, in part by the Fundamental Research Funds for the Central Universities under Grant N2224004-01, Grant N2224002-23, in part by the Central Government Guides Local Science and Technology Development Foundation under Grant 2022JH6/100100055, in part by the Key Laboratory of Intelligent Manufacturing Technology (Shantou University), Ministry of Education under Grant 202109241, and in part by the 111 Project 2.0 under Grant B08015. (*Corresponding Authors: Jinliang Ding, Gary G. Yen*)

Q. Chen, J. Ding, and T. Chai are with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, 110819, China (e-mail: cqd0309@126.com; jlding@mail.neu.edu.cn, tychai@mail.neu.edu.cn).

G. G. Yen is with the school of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74074, USA (gyen@okstate.edu).

S. Yang is with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, 110819, China, and also with the Centre for Computational Intelligence. School of Computer Science and Informatics, De Montfort University, Leicester, LE1 9BH, U. K. (syang@dmu.ac.uk).

This paper has supplementary material.

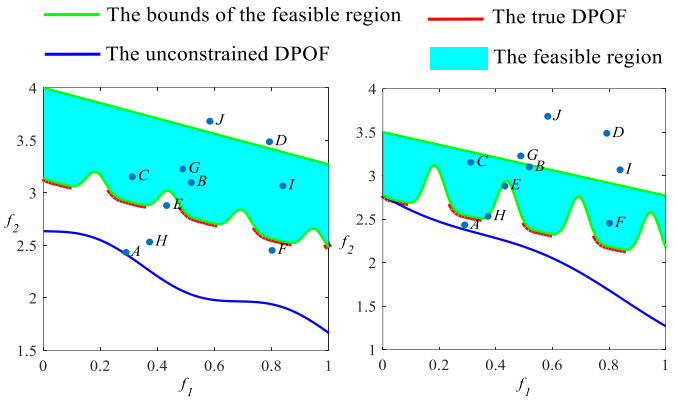

Fig. 1. The diagram of the true DPOFs of a DCMOP under t=1 (left) and t=2 (right), and A-J denote solutions with different objective values.

the DPOSs. For example, as shown in Fig. 1, the feasible region at t=1 is larger than that at t=2, while the range of the DPOF at t=1 is wider than that of the DPOF at t=2.

2) The number of infeasible solutions fluctuates with the feasible region, and irregular boundaries of the dynamic feasible region diversify the evaluation values of infeasible solutions. For example, the constraint violation (CV) value of solution E in Fig. 1(a) is larger than that of solution E in Fig. 1(b), but solution E in Fig. 1(a) is more likely to become a nondominated solution.
3) The optimal values of the distance variables in DPOSs move in different directions and magnitudes under dynamic environments, with some showing drastic changes, while others may move in small magnitudes, complicating the movement directions and magnitudes of the optimal distance variables.

Solving DCMOPs with the above characteristics may face the following challenges:

1) The challenges of handling and tracking infeasible solutions that may affect DPOFs. Infeasible solutions during evolution have different impacts on searching for the approximate DPOSs. For example, infeasible solutions far from the DPOFs may slow the population convergence, but small adjustments for some infeasible solutions (e.g., solutions E and F in Fig. 1(a)) may help them close to the true DPOSs, especially for infeasible solutions between the unconstrained DPOFs and the true DPOFs. Therefore, selecting infeasible solutions with good performance to update the population and track the true DPOFs is a difficult task.
2) Difficulties in designing dynamic response strategy. Optimizing different distance variables causes the solutions to move different distances toward the true DPOSs due to diverse movement directions and magnitudes of their optimal values. Therefore, it is difficult to design an effective dynamic response strategy to optimize solutions and reobtain a promising initial population.
3) Different characteristics of the true DPOFs are caused by the feasible region, and diverse changes of the optimal distance variables have impacts on the capability of an algorithm to track DPOFs. Therefore, designing scalable test suites with the above characteristics is arduous.

In recent years, there has been considerable success in solving constrained multiobjective optimization problems (CMOPs) and DMOPs, respectively, including constraint handling approaches [8] and dynamic response strategies [72]. However, there is limited literature on solving DCMOPs. Thus far, only Azzouz *et al*. [4], [5] and Chen *et al*. [2] have attempted to solve DCMOPs. Meanwhile, the optimal values of the distance variables in the benchmarks designed by Azzouz *et al*. [5] show only small magnitudes between two adjacent environments, and the test suites proposed by Chen *et al*. [2] do not consider how to simulate the diverse movement directions and magnitudes of the optimal distance variables when an environment changes. Additionally, the above algorithms can search for promising DPOSs with the same change in the optimal distance variables, but they perform poorly in solving DCMOPs with diverse changes.

To solve the above problems, this paper proposes a multi-population evolution based dynamic constrained multiobjective optimization algorithm (*m*EDCMOA in short) in which the population is divided into different tribes, each of which contributes to tracking the true DPOFs. Additionally, a scalable generator is designed to simulate diverse changes in the optimal distance variables under dynamic environments. The specific contributions of this paper are summarized as follows:

1) To take full advantage of solutions with different performances, especially solutions close to the DPOFs, this paper proposes a tribe classification operator to classify solutions into different tribes according to their feasibility and objective values, driving the population toward the true DPOFs.
2) This paper designs a population selection operator to update the population according to the feasible solutions as well as promising infeasible solutions that are identified from infeasible tribes by domination operators, improving the diversity of the population and the optimization efficiency of the algorithm.
3) To respond to diverse changing environments, we propose a dynamic response strategy to estimate the directions and magnitudes of the distance variables moving toward the new DPOFs in different tribes and update all solutions based on the evaluated values, regenerating an initialized population close to the DPOFs quickly.
4) To ensure that the optimal distance variables in test problems of DCMOPs have consistent change characteristics with real-world problems, we design a scalable generator in which the optimal values of the distance variables have diverse movement directions and magnitudes when dynamic environments change, testing the capability of the algorithm to track the DPOSs.

The remainder of this paper is organized as follows. Section II introduces some related works on constraint handling techniques and dynamic response strategies. Section III presents the details of the proposed *m*EDCMOA. Section IV discusses the designed test generator that can show diverse movement directions and magnitudes of the optimal distance variables. The experimental setup and experimental results of *m*EDCMOA compared with six state-of-the-art dynamic constrained multiobjective evolutionary algorithms (DCMOEAs) on test problems are given in Sections V and VI, respectively. Finally, Section VII outlines a summary and suggests research directions for future work.

II. Related Work

This section first gives problem formulation, followed by the existing methods for dynamic constrained optimization problems (DCOPs), which mainly focus on combining constraint handling techniques and dynamic response strategies [7]. Subsequently, we review differences between dynamic constrained single-objective and multiobjective approaches. The related benchmarks are discussed in the last paragraph.

A. Problem Formulation

Without loss of generality, a DCMOP can be mathematically formulated as follows [2]:

$$\begin{aligned} &\min\ F(\mathbf{x},t)=(f_1(\mathbf{x},t), f_2(\mathbf{x},t),..., f_m(\mathbf{x},t))^T \\ &subject\ to \begin{cases} h_k(\mathbf{x},t)=0, & k=1,...,H \\ g_k(\mathbf{x},t)\le 0, & k=H+1,...,G \\ \mathbf{x}\in\Omega \end{cases} \end{aligned} \tag{1}$$

where $\mathbf{x}=(x_1,\ldots,x_D)^T$ belongs to feasible space which is a subset of the search space $\Omega=[L_1, U_1]\times\cdots\times[L_j, U_j]\times\cdots\times[L_D, U_D]$, and L_j and U_j are the lower and upper bounds of the jth decision variable, respectively. t is the discrete environment index. $F(\mathbf{x},t)$ is a vector that includes m dynamic objective functions, and $\mathbf{x}$ is a vector that consists of D decision variables from the space Ω. $h(\mathbf{x},t)$ and $g(\mathbf{x},t)$ are the dynamic equality and inequality constraint functions, respectively, that vary with t. There are H equality and G-H inequality constraint functions, which must be satisfied by the optimal solution.

B. Constraint Handling Techniques

According to [8], many constraint handling techniques have been proposed to solve CMOPs, such as penalty function strategies, methods to separate objectives and constraints, multiobjective formulation, methods to transform CMOPs, hybrid approaches, methods of altering the reproduction operators, and other methods. These techniques will be discussed in the following subsections.

1) *Penalty Function Strategies*: These strategies construct the penalty term based on the *CV* values and add this term to objective functions. Woldesenbet *et al*. [9] designed a distance measure based on the original objective and *CV* values to guide the evolution of the population. Maldonado and Zapotecas-Martinez [10] proposed a dynamic penalty function strategy that changes parameters with the generation number. Yu *et al*. [11] proposed a dynamic selection preference-assisted strategy in which the selection preference of each solution is transformed between objectives and constraints. Ma and Wang [12] designed a shift-based penalty strategy to construct new objective functions with shift and penalty measures.

2) *Methods to Separate Objectives and Constraints*: Separation methods evaluate pairwise solutions by comparing their objective and *CV* values separately, mainly including the constrained-domination principle (CDP) proposed by Deb *et al*. [13], ε constrained method designed by Takahama and Sakai [14], stochastic ranking (SR), and so on. The CDP compares paired solutions A and B, and A is supposed to dominate B if 1) A is a feasible solution while B is an infeasible solution; 2) both A and B are infeasible solutions, and $CV(A)\le CV(B)$; and 3) both of them are feasible solutions, and $A\prec B$. Clearly, the CDP is a relatively simple method, and it has been applied in different constrained multiobjective evolutionary algorithms (CMOEAs) (e.g., MOSES [15], NSGA-III [16]). Inspired by the CDP, Fan *et al*. [17], [18] integrated angle information into the CDP (ACDP), further exploiting the information hidden from infeasible solutions.

The ε constrained methods use a parameter ε to relax constraints. A solution is considered a feasible solution if its *CV* value is less than ε, and A is supposed to dominate B if: 1) $CV(A)\le\varepsilon$, $CV(B)\le\varepsilon$, and all objective values of A are better than those of B; 2) $CV(A)\le\varepsilon$ and $CV(B)>\varepsilon$; or 3) $CV(A)>\varepsilon$, $CV(B)>\varepsilon$, and $CV(A)<CV(B)$. To solve CMOPs, the ε constrained method is embedded into different algorithm frameworks (e.g., NSGA-II [19], a multiobjective particle swarm optimization algorithm based on decomposition [20], and MOEA based on decomposition (MOEA/D) [21]) to handle constraints.

The SR method adopts a probability parameter *pf* to control the comparison of two individuals. It compares two individuals according to their objective values with the probability *pf*, and according to their *CV* values with the probability 1-*pf*. Jan and Khanum [22] embedded this method into the MOEA/D framework. Liu *et al*. [23] designed indicator-based CMOEAs by combining the indicator-based MOEA with the CDP, ε constrained method, and SR method.

For the above methods, the CDP usually gives priority to feasible solutions, accelerating the convergence rate of the population. However, this method may cause the population to fall into a local optimum. The ε constrained method and SR consider the information of infeasible solutions and can make up for the disadvantages of the CDP to a certain extent, but it is challenging to set ε and *pf* [8].

3) *Multiobjective Formulation*: Multiobjective methods regard constraints as one or more additional objective functions. In other words, a CMOP is transformed into an unconstrained MOP. Vieira *et al*. [24] transformed the constraints into two new objectives: one is the number of violations, and the other is calculated according to the penalty function. Long [25] regarded the convergence, diversity, and feasibility of solutions as three new objectives. By treating the *CV* as an objective, Peng *et al*. [26] designed a novel constraint handling technique based on directed weights to deal with CMOPs. Zhou *et al*. [27] proposed a tri-goal evolutionary framework that designs two indicators for diversity and convergence and transforms constraints into the third feasibility indicator.

4) *Methods to Transform CMOPs*: To efficiently solve CMOPs, many researchers transform CMOPs into other problems (e.g., two-stage optimization problems). Subsequently, some promising operators are adopted to explore the search region. For example, Wang *et al*. [28] designed a cooperative differential evolution framework that includes m subpopulations, each of which optimizes an objective with constraints. Yang *et al*. [29], Tian *et al*. [30], and Wang *et al*. [31] converted a CMOP into a two-population optimization problem, and information and knowledge of two populations are interacted and transferred between them. Li *et al*. [32] designed a two-archive evolutionary algorithm in which one aims at searching for the population near the POF, while the other is used to explore more regions and maintain population diversity. Liu *et al*. [33] designed a bidirectional coevolution algorithm and adopted the main population to maintain feasibility and the archive population to maintain diversity.

Santana-Quintero *et al*. [34] designed a two-stage

optimization in which MOEA pushes the population close to the POF in the first stage, and fuzzy set theory improves the population diversity and the convergence in the second stage. Fan *et al*. [35] and Ming *et al*. [36] proposed a push and pull search framework in which the population can across the infeasible region to the unconstrained POF in the pushing stage, and the ε constrained method is used to search for the constrained POF. Tian *et al*. [37] and Liu and Wang [38] presented a two-stage evolutionary algorithm that can help the population search for the feasible region in one stage and spread along the feasible boundary in the other stage. In [39], the optimization process is divided into two stages that consider the convergence and diversity in the first stage and maintain population feasibility and diversity in the second stage. Liang *et al*. [40] developed a two-stage method that exploits a learning stage to obtain the relationship between the constrained and unconstrained POF and then uses this information to guide the use of reasonable evolutionary strategies in the evolving stage.

5) *Hybrid Approaches*: To handle CMOPs with equality constraint functions more effectively, some researchers combined EAs with mathematical programming that mainly focuses on improving the local search capability. For example, Kelner *et al*. [41] combined a genetic algorithm with a local search strategy based on the interior point method. Datta *et al*. [42] merged the nonconvex radial boundary intersection with an interior point method. Uribe *et al*. [43] hybridized MOEAs with a descent direction calculation method. Hernandez *et al*. [44] designed an approach of mixing the evolutionary strategy and hypervolume Newton method.

6) *Methods of Altering Reproduction Operators*: These methods focus on the design of reproduction operators. For example, Yu *et al*. [45] proposed a new mutation mechanism to handle infeasible and feasible solutions, and He *et al*. [46] emphasized the role of offspring generation when reproducing promising feasible or useful infeasible offspring solutions and designed a self-adaptive differential evolution with Pareto dominance, in which the parameters are adjusted adaptively.

7) *Other Methods*: Some methods do not belong to the above categories, and they are slotted into this class. For example, Datta and Regis [47] proposed a surrogate-assisted evolution strategy to solve CMOPs with expensive black-box objective functions subjected to expensive inequality constraint functions. Qu and Suganthan [48] adopted multiple constraint handling techniques to solve CMOPs. Zapotecas-Martinez and Coello Coello [49] proposed a novel algorithm for handling continuous box-constrained multiobjective optimization problems in which a nonlinear simplex search strategy is adopted to obtain multiple solutions of the POS.

Indeed, some designs of the constraint handling techniques in solving CMOPs can be extended in solving DSCOPs. For example, penalty function strategies [50], multiobjective methods [51], and hybrid methods [52] have been proposed to solve DSCOPs. These constraint handling techniques show different advantages in solving different CMOPs and DSCOPs, and more discussions about these techniques for solving CMOPs and DSCOPs can be found in [8] and [85], respectively. However, for a DCMOP, the infeasible and feasible solutions change drastically under diverse environments, altering the advantages/deficiencies of different solutions in driving the population toward the DPOSs. It is difficult for a single strategy to deal with all infeasible solutions effectively. Therefore, designing different evolution operators to handle infeasible solutions with varying performances is beneficial for the population to approach the true DPOFs.

C. Dynamic Response Strategies

The strategies for responding to dynamic changes can be divided into four categories: diversity enhancement methods, multi-population designs, memory-based approaches, and prediction-based techniques.

1) *Diversity Enhancement*: In recent years, researchers have proposed several diversity enhancement methods to reinitialize the population to approximate the new POS [53]-[58]. For example, to maintain population diversity, Yang and Tinos [53] and Mavrovouniotis and Yang [55] proposed a hybrid immigrant scheme including memory-based immigrants [54], random immigrants, and elitism-based immigrants. Ruan *et al*. [56] and Ma *et al*. [57] proposed diversity maintenance methods to help the reinitialized population adapt to the new POS. A high level of population diversity is maintained in [58] by preserving the solutions with more uniform distances.

2) *Multi-population Designs*: Multi-population designs adopt multiple subpopulations to explore regions after detecting various environments. Goh and Tan [59] proposed to compare the potential of new regions with past information to decide whether a subpopulation should be initialized when an environmental change occurs. Namely, the particular subpopulation must be reinitialized in the space from which the winner is sampled. Gong *et al.* [60] suggested a cooperative coevolutionary optimization framework that divides decision variables into two cooperative subpopulations according to the interval similarity between decision variables and interval parameters. In [61], Liang *et al.* divided decision variables into two subpopulations in the static stage and three subpopulations in the dynamic response stage, maintaining the balance between population diversity and convergence.

3) *Memory-based Approaches*: These approaches reuse the past and relevant information to improve the population performance when a periodical change occurs [62]. For example, Chen *et al.* [2], Zhang *et al*. [63], Jiang and Yang [64], and Woldesenbet and Yen [65] reused a portion of historical solutions to regenerate solutions close to new POFs after environments vary. In [66], Azzouz *et al*. designed a dynamic response method based on a memory-based strategy that uses previous solutions. Liang *et al*. [67] analyzed the change similarity to the historical ones and designed two response strategies.

4) *Prediction-based Techniques*: Such techniques usually predict new optimal solutions using existing information and other learning techniques. Recently, prediction-based techniques that exploit different strategies (e.g., transfer learning [68]-[72], reinforcement learning [73], [74], multidirectional prediction approaches [75], support vector regression [76], [77], mixture-of-experts [78], the multimodel prediction method [79], an efficient self-adaptive precision controllable operator [80] and a knee-guided prediction approach [81]) to predict where the population should be reinitialized have received much attention. Transfer learning-based methods solve DMOPs by transferring or

sharing knowledge according to domain-specific similarities.

Some strategies similar to the above mentioned approaches are also used for solving dynamic constrained single-objective optimization problems (DCSOPs). For example, the diversity enhancement method [82], multipopulation design [83], modified memory strategy [84], and their ensemble strategies [85] were designed to respond to dynamic environments in DCSOPs. These change response strategies perform well for solving DMOPs and DCSOPs, and additional discussions regarding similar strategies for solving DMOPs and DSCOPs can be found in [72] and [85], respectively. However, for DCMOPs with diverse movement directions and distances of the optimal distance variables, some variables require substantial adjustments, while other variables require only small adjustments. Therefore, dynamic constraint information and diverse characteristics of the optimal distance variables need to be embedded in the dynamic response strategy, regenerating a promising population closer to the DPOSs.

D. Differences between Dynamic Constrained Single-Objective and Multiobjective Approaches

In a DCSOP, a solution is better than another solution if the objective value of the former is better than the latter. Unlike the DCSOP, comparing two solutions is not as straightforward in the DCMOP, and Pareto dominance is most commonly used. Therefore, dynamic constrained single-objective optimization algorithms (DCSOAs) encourage the search toward an optimal solution, while DCMOEAs promote the search to move closer to the Pareto fronts and maintain diversity in the population. In other words, the main difference between the ideas of solving DCMOPs and DCSOPs is the fitness evaluation process. Notably, this evaluation process is mainly applied to the mating selection and population update operators. In DCSOPs, these two operators usually choose solutions with promising objective and *CV* values. However, the evaluation in DCMOPs is more complicated. First, ranks are assigned to solutions according to the designed dominant relationship. Then, a fitness value is assigned to each solution based on its ranking.

E. Survey of Existing Benchmarks on Related Works

Benchmarks play a crucial role in judging whether an algorithm is a candidate for solving multiobjective optimization problems (MOPs) [91]. Currently, benchmarks for DMOPs [91] and CMOPs [16] have been proposed. The above research plays a certain role in promoting the DCMOP test suite. However, these test problems do not simultaneously involve dynamism and constraints, resulting in the lack of problem characteristics arising in DCMOPs. Azzouz *et al*. [5] directly embedded dynamic parameters into static CMOPs, obtaining a set of benchmarks for DCMOPs with small movement magnitudes of the optimal distance variables. Chen *et al*. [2] designed a set of test problems for DCMOPs that consider simultaneous changes in the feasible region and unconstrained DPOF. However, there are many industrial problems (e.g., the operation optimization problem of the catalytic cracking process solved in [1]) in which the optimal distance variables have diverse movement directions and distances. Therefore, it is difficult to test the tracking capability of the algorithm if this characteristic is ignored.

Algorithm 1: Basic Framework of *m*EDCMOA

```
1:  Input: P=Φ, AS=Φ
2:  Output: Approximate nondominated solutions
3:  Initialization: generate an initial population P with N random solutions;
4:  while termination condition is not met do
5:    if change is detected then
6:      Objective Function Calculation (P, AS)
7:      P=Dynamic Response Operator (P);
8:      AS=Nondominated Solution Selection Operator (P, AS);
9:    end if
10:   P'=Mating Selection Operator (P);
11:   Q= Offspring Generation Operator (P');
12:   (T1, T2…)=Tribe Classification Operator (P, Q);
13:   P=Population Selection Operator (T1, T2…);
14:   AS= Nondominated Solution Selection Operator (P, AS);
15: end while
```

III. PROPOSED *M*EDCMOA

This section first presents the basic framework of the proposed *m*EDCMOA, followed by the details of the building blocks in this algorithm. Finally, we analyze the computational complexity of the proposed algorithm.

A. Framework of mEDCMOA

The basic framework is presented in Algorithm 1. For a given DCMOP, an initial population *P* with *N* random solutions is generated. In each generation, until the termination condition is met (line 4 of Algorithm 1), *m*EDCMOA first detects whether there is an environmental change (line 5 of Algorithm 1). The dynamic response operator is used to update *P* once a changed environment is detected (lines 6–7 of Algorithm 1). Afterward, the nondominated selection operator identifies nondominated solutions from *P* and the nondominated solution set *AS* (line 8 of Algorithm 1). Subsequently, *m*EDCMOA adopts the mating selection operator to obtain a parent population *P'* and adopts the genetic operators to generate an offspring set *O* (lines 10–11 of Algorithm 1). Based on the above results, the tribe classification operator divides *P* and *O* into different tribes T_1, T_2…, (line 12 of Algorithm 1). Finally, the population selection strategy is applied to different tribes to update *P*, and the nondominated solution selection operator is adopted to update *AS* according to *P* and the previous *AS* (lines 13–14 of Algorithm 1). The above steps (lines 5–14) are repeated until the termination condition is met.

B. Initialization

The initial population *P* includes at least one feasible solution to ensure that the initial DPOSs and DPOFs are not empty. The initial population is generated as follows.

The *j*th variable of the *i*th solution is generated as follows:

$$x_{i,j} = L_j + rand \times (U_j - L_j) \tag{2}$$

where *rand* is a random number generator belonging to [0, 1].

Subsequently, the *CV* value, $CV(\mathbf{x}_i, t)$, of the *i*th solution $\mathbf{x}_i$ against *H* equality and *G-H* inequality constraint functions at time *t* is calculated using Equations (3) and (4),

$$CV(\mathbf{x}_i, t) = \sum_{k=1}^{H} c_k(\mathbf{x}_i, t) \tag{3}$$

$$\text{where } c_k(\mathbf{x}_i, t) = \begin{cases} \max\left(0,\ |h_k(\mathbf{x}_i, t)| - \delta\right) & k = 1, \ldots, H \\ \max\left(0,\ g_k(\mathbf{x}_i, t)\right) & k = H+1, \ldots, G \end{cases} \tag{4}$$

Algorithm 2: Tribe classification operator

```
Input: population, FT=Φ, IT=Φ, DIT=Φ, NIT=Φ,
Output: FT, DIT, NIT.
Calculate CV values of solutions in population.
For i=1:N
  if CV(x_i, t)>0 then
    Push x_i into the infeasible set IT;
  else
    Push x_i into the feasible set FT;
  end if
end for
Update nondominated solutions set AS according to FT and AS
For i=1: |IT|;
  if the ith infeasible solution is not dominated by solutions in AS then;
    Push the ith infeasible solution into DIT;
  else
    Push the ith infeasible solution into NIT;
  end if
end for
```

where δ is the tolerance value of equality constraint functions.

If $CV(\mathbf{x}_i, t)$ is equal to zero, then $\mathbf{x}_i$ is a feasible solution; otherwise, it is an infeasible solution. If the initial population has at least one feasible solution, the algorithm starts iterative optimization; otherwise, the initial population is regenerated.

C. Tribe Classification Operator

For a given DCMOP with minimal objective functions, some solutions have small objective values, but they may be infeasible due to the presence of constraints. For example, in the dynamic optimization of fluid catalytic cracking-distillation processes, there are some infeasible solutions close to the unconstrained DPOF when the lower bounds of fossil fuel yield constraints are large. In fact, some infeasible solutions (e.g., solutions *A*, *E*, and *F* in Fig. 1(a)) may help feasible solutions with large objective values (e.g., solution *I* in Fig. 1(a)) track the DPOFs, pushing other infeasible solutions (e.g., solution *J* in Fig. 1(a)) into the feasible region or close to the DPOFs. To identify infeasible solutions with such potential, this paper proposes a tribe classification operator to classify the population into different tribes according to the feasibility check and objective values. Specifically, the population is first divided into feasible and infeasible tribes (*FT* and *IT*, respectively) according to their feasibility, and we exploit the nondominated solution selection operator to update *AS* from *FT*. Note that any nondominated solution selection method can be used here. Then, some infeasible solutions with large objective values (e.g., solutions *J* and *D* in Fig. 1(a)) are usually dominated by nondominated solutions and may have difficulty driving the population near the DPOFs. Therefore, this paper proposes classifying infeasible solutions into different tribes according to whether they are dominated by nondominated solutions. Namely, an infeasible solution $\mathbf{x}$ is called a nondominated infeasible solution if its objective values are not worse than the objective values of nondominated solutions, otherwise, $\mathbf{x}$ is considered a dominated infeasible solution. Based on the above definition, the infeasible solutions are divided into dominated and nondominated infeasible tribes (*DIT* and *NIT*, respectively). The pseudocode of the tribe classification operator is presented in Algorithm 2.

As an illustrative example, this paper gives the objective and *CV* values of ten solutions shown in Fig. 1(a), as listed in Table I. In this example, solutions *B*, *C*, *G*, and *I* belong to *FT* because their *CV* values are zero, while solution *G* is dominated by solution *C*. Therefore, solutions *B*, *C*, and *I* are nondominated solutions. Solutions *A*, *E*, *F*, and *H* belong to *NIT* because their objective values are better than those of solutions *B*, *C*, and *I*. Solutions *D* and *J* belong to *DIT* because they are infeasible and dominated by nondominated solutions.

TABLE I
Objective and *CV* values of Ten Solutions at t=1

Solution No.	*A*	*B*	*C*	*D*	*E*
f_1	0.28929	0.51792	0.31085	0.79181	0.43147
f_2	2.43310	3.09818	3.15310	3.48583	2.87879
$CV(\mathbf{x}_i,t)$	0.93536	0	0	0.08742	0.12456
Solution No.	*F*	*G*	*H*	*I*	*J*
f_1	0.80099	0.48777	0.37192	0.83896	0.58310
f_2	2.45233	3.22608	2.53142	3.06631	3.68127
$CV(\mathbf{x}_i,t)$	0.14786	0	0.67842	0	0.15489

Algorithm 3 Mating selection operator

```
Input: Population P, P'=Φ
Output: Parent P'
Calculate the modified objective values of each solution in P.
Calculate the fitness value of each solution according to the modified
objective values.
Calculate the crowding distances of all solutions by using the method
proposed in NSGA-II.
for i=1:N
  Select two random individuals p_1, p_2 from P;
  if p_1 ≺ p_2 then
    Push p_1 into P';
  else
    if p_2 ≺ p_1 then
      Push p_2 into P';
    else
      if d(p_1)>d(p_2)
        Push p_1 into P';
      else
        if d(p_1)<d(p_2) then
          Push p_2 into P';
        else
          Select a random individual from p_1 and p_2, and push it into P';
        end if
      end if
    end if
  end if
end for
```

D. Mating and Population Selection Operators

Mating selection is essential in producing new offspring because it is the key to determining whether the population can search for solutions close to the DPOFs. DCMOPs involve dynamic constraints, implying that the parent population may include infeasible solutions. For example, in the dynamic optimization of fluid catalytic cracking-distillation processes described in [1], the number of feasible solutions is less when the change range of yield constraints is small, and some infeasible solutions are used to update the population. The mating selection method proposed in [2] has shown good capability in balancing infeasible and feasible solutions, and it is incorporated into this work to choose the parent population. The pseudocode for the mating selection operator is presented in Algorithm 3. Note that this selection method adopts the penalty function method proposed by Woldesenbet *et al.* [9] to modify the objective values according to *CV* values and uses the crowding distance calculation operator proposed in [13] to calculate crowding distances (*d*). The procedures of the penalty function method and the crowding distance calculation operator

are discussed in Sections S-II and S-III of the Supplementary Material, respectively.

This paper adopts the popular simulated binary crossover (SBX) and polynomial mutation (PM) operators proposed in [88] to generate offspring once the parent population is constructed. The procedure of the offspring generation operator is presented in Section S-IV of the Supplementary Material. Subsequently, we design a population selection operator to update the population based on the problem characteristics. Specifically, for a DCMOP, *m*EDCMOA needs to have a good convergence capability so that solutions can track the true DPOFs quickly under diverse dynamic environments. Indeed, feasible and infeasible regions decrease or increase with dynamic environments, changing the number of solutions in different tribes. The number of feasible solutions in the population is large when the feasible region increases, and reserving feasible solutions would encourage algorithm convergence. Meanwhile, reserving a portion of the more promising infeasible solutions is beneficial for increasing population diversity. To balance the convergence speed of the algorithm and the diversity of the population, this paper proposes a population selection operator in which some nondominated and dominated infeasible solutions may be used to update the population if the number of feasible solutions is not enough. The details of updating the population are described as follows.

First, several tribes (i.e., *FT*, *IT*, *DIT*, and *NIT*) are cleared, and the tribe classification operator is adopted to classify solutions in *P* and *Q* into *FT*, *DIT*, and *NIT*. All feasible solutions are copied to an empty set *NP*.

Second, if the number of nondominated infeasible solutions is more than *N*-|*FT*|, then some nondominated infeasible solutions are pushed into *NP*, while dominated infeasible solutions are not considered. Considering that small adjustments in the infeasible solutions close to the DPOFs (e.g., solution *F* in Fig. 1(a)) may help them become feasible or even nondominated solutions, we propose a domination operator to choose *N*-|*FT*| nondominated infeasible solutions from *NIT*. Specifically, in a DCMOP with minimal objective functions, the solutions with the large objective values in *NIT* are closer to the true DPOF. Therefore, this paper designs a fitness calculation method for nondominated infeasible solutions, which is given in Equation (5). Each solution $\mathbf{x}_i$ in *NIT* is associated with a fitness value $F'(\mathbf{x}_i)$ that does not consider *CV*.

$$F'(\mathbf{x}_i) = \left|\left\{\mathbf{x}_{i'} \in NIT \mid \nexists \mathbf{x}_i,\ \mathbf{x}_{i'} \prec \mathbf{x}_i\right\}\right| \tag{5}$$

where $|\cdot|$ represents the cardinality of $\mathbf{x}_{i'}$ dominating $\mathbf{x}_i$. $\mathbf{x}_{i'} \prec \mathbf{x}_i$ indicates that $\mathbf{x}_{i'}$ dominates $\mathbf{x}_i$ if and only if $f_m(\mathbf{x}_{i'}, t) \le f_m(\mathbf{x}_i, t)$ for every m and $f_b(\mathbf{x}_{i'}, t) < f_b(\mathbf{x}_i, t)$ for at least one index $b \in \{1,\ldots,m\}$.

We use the solutions shown in Fig. 1 and Table I as an example to illustrate the fitness calculation method for Equation (5). Solutions *A*, *E*, *F*, and *H* belong to *NIT*, and the fitness values of solutions *E* and *F* are zero. The fitness value of solution *H* is one because $f_m(\mathbf{x}_H, t) < f_m(\mathbf{x}_E, t)$ for each $m \in \{1, 2\}$. Solution *A* has a fitness value of three because $f_m(\mathbf{x}_A, t) < f_m(\mathbf{x}_E, t)$, $f_m(\mathbf{x}_A, t) < f_m(\mathbf{x}_F, t)$ and $f_m(\mathbf{x}_A, t) < f_m(\mathbf{x}_H, t)$ for each $m \in \{1, 2\}$.

After the fitness values of nondominated infeasible solutions are calculated according to Equation (5), this method selects *N*-|*FT*| nondominated infeasible solutions according to their fitness values and crowding distances, pushing them into *NP*.

Algorithm 4. Population selection strategy

```
Input: P, Q, IF=Φ, DIT=Φ, NIT=Φ, FT=Φ, NP=Φ
Output: A new population NP;
Adopt the tribe classification operator to obtain FT, DIT, and NIT.
if feasible solutions do not fill the population then
    All feasible solutions are copied to an empty set NP.
    if the number of nondominated infeasible solutions is larger than N-|FT| then
        Calculate the fitness values of nondominated infeasible solutions using Equation (5).
        Choose N-|FT| nondominated infeasible solutions according to their fitness values and crowding distances, and push them into NP.
    else.
        Push all nondominated infeasible solutions into NP.
        Calculate the fitness values of dominated infeasible solutions using Equation (6).
        Choose N-|FT|-|NIT| dominated infeasible solutions according to their fitness values and crowding distances, and push them into NP.
    end if
else
    Calculate the fitness values of feasible solutions according to Equation (6).
    Choose N solutions according to their fitness values and crowding distances, and push them into NP.
end if
```

It is worth noting that if the number of nondominated infeasible solutions is less than *N*-|*FT*|, then *N*-|*FT*|-|*NIT*| dominated infeasible solutions are selected and pushed into *NP*. In this case, the population selection operator does not calculate the fitness values of nondominated infeasible solutions, and all nondominated infeasible solutions are pushed into *NP*. To choose *N*-|*FT*|-|*NIT*| dominated infeasible solutions close to the DPOFs, this method adopts Equation (6) to calculate the fitness value $F(\mathbf{x}_i)$ of a dominated infeasible solution $\mathbf{x}_i$ and selects *N*-|*FT*|-|*NIT*| dominated infeasible solutions from *DIT* according to their crowding distances, pushing them into *NP*.

$$F(\mathbf{x}_i) = \left|\left\{\mathbf{x}_{i'} \in R \mid \nexists \mathbf{x}_i, \mathbf{x}_{i'} \prec \mathbf{x}_i\right\}\right| \tag{6}$$

where *R* is the *FT* or *DIT*. $\mathbf{x}_{i'} \prec \mathbf{x}_i$ indicates that $\mathbf{x}_{i'}$ dominates $\mathbf{x}_i$ if and only if $f_m(\mathbf{x}_{i'}, t) \le f_m(\mathbf{x}_i, t)$ for every m and $f_b(\mathbf{x}_{i'}, t) < f_b(\mathbf{x}_i, t)$ for at least one index $b \in \{1,\ldots,m\}$.

Additionally, a large number of feasible solutions means that the feasible space is large or the evolution process may be in the late stage, and infeasible solutions are not considered. This paper adopts Equation (6) to calculate the fitness values of feasible solutions and select *N* solutions, pushing them into *NP*.

The pseudocode for the population selection strategy is presented in Algorithm 4.

E. Dynamic Response Strategy

Change detection is usually needed before responding to environmental change. The authors of [2], [59], and [64] designed different detection strategies, and these strategies can be used to detect dynamic environments in DCMOPs. This paper focuses on only the proposed dynamic response strategy.

A real-world DCMOP may show different dynamic characteristics such as irregular changes in the feasible region, disconnected DPOFs, and changes in the optimal distance variables in a dynamic environment. Solutions in different tribes need to move different directions and distances before they become nondominated solutions. For example, in an objective space, the objective values of most nondominated infeasible solutions (e.g., solutions *E* and *H* in Fig. 1(a)) need to

be increased before they become nondominated ones, while the objective values of dominated infeasible solutions (e.g., solution *D* in Fig. 1(a)) may have to be decreased. Therefore, it is necessary to adopt different adjustment directions and magnitudes to optimize the distance variables of solutions in different tribes, improving the speed of regenerating an initial population close to the relocated DPOFs.

During the evolution process, the true DPOFs or infeasible region may change over *t*. The previous nondominated solutions may have become dominated or even infeasible ones. Similarly, the previous infeasible solutions may have become feasible or nondominated ones. To explore new feasible subregions, this paper proposes a dynamic response strategy to modify the previous and random solutions in different tribes to regenerate an initial population when a changed environment arises. The detailed method is described as follows.

First, *m*EDCMOA generates half of *N* solutions randomly in the changing environment by the initialization method and recalculates the objective and *CV* values of previous solutions. All solutions (i.e., random and previous solutions) are pushed into different tribes by the tribe classification operator.

Second, considering that the optimal values of the distance variables moving to the true DPOSs are different, we design an adjustment evaluation method based on a one-dimensional search strategy to estimate the movement directions and magnitudes of the distance variables in each tribe. Specifically, this method calculates the fitness value of each solution and chooses solutions with a fitness value of zero in each tribe. Then, the adjustment direction and magnitude of each distance variable of solutions with a fitness value of zero in each tribe are determined. For a solution with a fitness value of zero in *FT*, a positive value δ is added to one dimension of this solution. If this decision variable is still in $[L_j, U_j]$ and at least one of the objective values of this solution is improved, then δ is continuously added to the corresponding dimension until this individual is no longer improved; Otherwise, the one-dimensional search strategy decreases by the value of δ into this dimension, and δ is continuously decreased in the corresponding dimension until this individual is no longer improved. The above one-dimensional search strategy is also used for infeasible solutions with a fitness value of zero, and the only difference is the evaluation method of an infeasible solution. For an infeasible solution, if its *CV* decreases or if it becomes a feasible solution, then this infeasible solution is considered to be improved, and δ is continuously increased or decreased in the corresponding dimension. The pseudocode for the one-dimensional search strategy for a solution with a fitness value of zero is shown in Algorithm 5.

To estimate the movement directions and magnitudes of the distance variables in different tribes, all variable changes and the number of improved solutions in the corresponding dimension are recorded, and the movement direction and magnitude of the *j*th distance variable in the *s*th tribe are calculated as follows.

$$v_{s,j} = \frac{tv_{s,j}}{tn_{s,j}} \tag{7}$$

where *s* is the tribe index (*s*=1 refers to *FT*, *s*=2 refers to *NIT*, and *s*=3 refers to *DIT*), and $tv_{s,j}$ denotes the total change value of the *j*th distance variable in the individuals with a fitness value of zero in the *s*th tribe. $tn_{s,j}$ is the number of improved solutions in the *j*th distance variable in the *s*th tribe. $v_{s,j}$ represents the movement direction and magnitude of the *j*th distance variable in the *s*th tribe, and the sign of $v_{s,j}$ denotes the direction.

Algorithm 5 One-dimensional search strategy

```
1:  Input: y=x_i, flag=0;
2:  Output: an improved x_i
3:  for j=1:D
4:    flag=0;
5:    do
6:      if flag=0 then
7:        y_j=x_{i,j}+δ;
8:      else
9:        y_j=x_{i,j}-δ;
10:     end if
11:     Calculate the objective values F(y,t) and CV(y,t) of y;
12:     if y_j≤U_j or y_j≥L_j
13:       if x_i is a feasible solution then
14:         if V(y,t)=0 and y dominates x_i then
15:           x_{i,j}=y_j
16:         else
17:           flag++;
18:         end if
19:       else
20:         if CV(y,t)≥CV(x_i,t) or y dominates x_i then
21:           x_{i,j}=y_j
22:         else
23:           flag++;
24:         end if
25:       end if
26:     else
27:       flag++;
28:     end if
29:   while (flag<2)
30: end for
```

After updating all solutions with a fitness value of zero, $v_{s,j}$ is added to the *j*th variable of other solutions in the *s*th tribe. However, some solutions near boundaries of the feasible region may become infeasible if $v_{s,j}$ is added to the corresponding variable. This paper adopts the following method to improve the corresponding solutions.

$$x_{i,j} = \begin{cases} x_{i,j} + rand \times (U_j - x_{i,j}) & if\ x_{i,j} + v_{s,j} > U_j \\ x_{i,j} + rand \times (L_j - x_{i,j}) & if\ x_{i,j} + v_{s,j} < L_j \end{cases} \tag{8}$$

where $x_{i,j}$ is the *j*th distance variable in the *i*th solution in a tribe.

The pseudocode for the dynamic response strategy is presented in Algorithm 6.

F. Computational Complexity of mEDCMOA

In the loop (lines 5–14 of Algorithm 1) of *m*EDCMOA, the most time-consuming operators include mating selection, offspring production, population selection, and nondominated solution selection, while other procedures involve less computational cost. For the mating selection operator, the time complexity is $O(mN)$, where *m* denotes the number of objectives and *N* is the population size. For the offspring production operator, generating an offspring solution requires $O(m)$ computations, so its time complexity is $O(mN)$. The population selection procedure requires $O(mN^2)$ computations on the fitness assignment and $O(N^2\log N)$ computations when updating the population. The nondominated solution selection operator executes $O(mN^2)$ computations. Thus, the total time complexity of *m*EDCMOA for one generation is $O(mN^2)$ or $O(N^2\log N)$, whichever is larger.

Algorithm 6 Dynamic response strategy

Input: P, $R=\Phi$, $FT=\Phi$, $DIT=\Phi$, and $NIT=\Phi$.
Output: P
Initialization: generate a population R with 50% of N solutions randomly;
Recalculate the objective and CV values of all solutions in P;
Divide solutions in P and R into FT, DIT, and NIT according to the tribe classification operator;
Calculate the fitness values of solutions in FT and DIT using Equation (6) and fitness values of solutions in NIT according to Equation (5);
Determine the distance and direction of the jth dimension variable in each tribe according to the one-dimensional search strategy and Equation (7).
for the ith solution in the sth tribe (i.e., FT, DIT, or NIT)
  if $F(i)$!=0 **then**
    for j=1:D
      $x_{i,j}=x_{i,j}+v_{s,j}$
      Repair $x_{i,j}$ by Equation (8) if it is less than L_j or greater than U_j;
    end for
    Calculate the objective and CV of $\mathbf{x}_i$;
  end if
end for
Combine all solutions in P, R and tribes.
Update the population with the population selection strategy according to merged solutions.

IV. Proposed Test Suite Generator

A. Designed Generator Diversifying the Optimal Distance Variables

There are many DCMOPs with diverse movement directions and distances of the optimal distance variables in industry. For example, in the catalytic-cracking optimization problem in [1], some optimal distance variables (i.e., the calorific values of heat recovery in the fractionator) change greatly, while the optimal values of the distance variables farther away from the fractionator (e.g., the flow rate of the stripping steam) change relatively little. In such a case, the dynamic response strategy needs to optimize different distance variables based on the distances they move to the optimal values.

Second, the optimal values of all distance variables have drastic change ranges when the production environment changes violently. For example, the fossil fuel yield constraints change greatly when the demands of gasoline, diesel, and natural gas change substantially, and almost all decision variables in the regenerator, reactor, and fractionator of the fluid catalytic cracking unit must be adjusted with large magnitudes. Indeed, production environments with drastic changes usually occur once in intervals. Therefore, some distance variables usually need large adjustments, while others are adjusted substantially after several intervals. Based on the above characteristics, the optimal values of several distance variables in the designed test suites usually change drastically after several environments.

To create a DCMOP with different changes in the optimal distance variables when dynamic environments vary, this paper designs the following generator based on exponential and trigonometric functions to ensure that the optimal values of the distance variables have different changes, which is as follows,

$$g(\mathbf{x},t)=\sum_{j=jl}^{D}\left(\begin{array}{l} x_j-\coprod_{j\in[jl,ju]}\left(\frac{c}{e^{\eta}}\times e^{t \bmod tl\times\lambda+\alpha\times\sin(j\times t\times\beta\times\pi)}\right)- \\ \coprod_{j\in[ju+1,D]}\left(k\times\sin(j\times(t+1)\times\beta\times\pi)+\left\lfloor\frac{t+\varepsilon}{tl}\right\rfloor\times\chi\right)\end{array}\right)^2 \quad (9)$$

where jl and ju control the number of the optimal distance variables with drastic changes when dynamic environments vary, and tl controls the frequency of occurrence of a drastic environment. c and η determine the changes in the optimal distance variables belonging to $[jl, ju]$, and k and χ control the changes in the optimal distance variables belonging to $[ju+1, D]$. ε denotes the number of slightly dynamic environments before the first drastic environment arises. β avoids the optimal values of the $(ju+1)$th to the dth distance variables being the same as their historical values. λ and α ensure that x_j is less than c, namely, $\eta\geq(t \bmod tl)\times\lambda+\alpha\times sin(j\times t\times\beta\times\pi)$, where $j\in[jl, ju]$.

Remark: The boundaries between great and small changes in the optimal distance variables in various real-world problems are different. For example, the calorific value of energy recovery in the distillation unit in [87] has seven orders of magnitude, and a change is considered small if this variable changes by two orders of magnitude. However, a change in the feedstock temperature of 10℃ is considered a great fluctuation. Therefore, boundaries between great and small changes in the optimal distance variables must be defined according to the actual production characteristics when using *m*EDCMOA to solve DCMOPs with diverse changes.

B. Test Problems

The DPOFs include different problem characteristics such as convexity–concavity and connectedness–disconnectedness, which can pose great challenges for DCMOEAs in tracking capability. This paper embeds the designed generator (i.e., Equation (9)) into the multiobjective and constraint functions of the test suites designed in [2], obtaining eight improved DCMOPs with different changes in the optimal distance variables and showing different characteristics of the DPOFs and feasible region. To highlight the influence of the distance variables on the objective values, this paper proposes that $(1+g(\mathbf{x},t))$ of all test problems is replaced by $(1+g(\mathbf{x_{II}},t))(1+g(\mathbf{x_{III}},t))$, and the expressions of $g(\mathbf{x_{II}},t)$ and $g(\mathbf{x_{III}},t)$ are as follows.

$$\begin{aligned} g(\mathbf{x_{II}},t)&=\sum_{j=jl}^{ju}\left(x_j-\left(\frac{c}{e^{\eta}}\times e^{t \bmod tl\times\lambda+\alpha\times\sin(j\times t\times\beta\times\pi)}\right)\right)^2 \\ g(\mathbf{x_{III}},t)&=\sum_{j=ju+1}^{D}\left(x_j-\left(k\times\sin(j\times(t+1)\times\beta\times\pi)+\left\lfloor\frac{t+\varepsilon}{tl}\right\rfloor\times\chi\right)\right)^2 \end{aligned} \quad (10)$$

where $\mathbf{x_{II}}=\{x_{jl}, x_{jl+1},\ldots, x_{ju}\}$, and $\mathbf{x_{III}}=\{x_{ju+1}, x_{ju+2},\ldots, x_D\}$. The optimal values of the distance variables in $\mathbf{x_{II}}$ have a drastic change when dynamic environments change, while only the optimal distance variables in $\mathbf{x_{III}}$ show a drastic change every tl dynamic environments.

V. Experimental Setup

A. Comparison Algorithms

For performance comparison, three state-of-the-art DCMOEAs (i.e., DC-MOEA [4], DC-NSGA-II-A [5], and *d*CMOEA [2]) and three modified CMOEAs (i.e., constrained NSGA-II (C-NSGA-II) [9], constrained NSGA-III (C-NSGA-III) [16] and constrained TAEA (C-TAEA) [32])) are considered. C-NSGA-II, C-NSGA-III, and C-TAEA are used to solve static multiobjective optimization problems. Therefore, C-NSGA-II, C-NSGA-III, and C-TAEA share the dynamic response scheme designed in [2] for responding to

dynamic environments, and they are called DC-NSGA-II, DC-NSGA-III, and DC-TAEA, respectively. All algorithms were coded using VC++6.0, and they were run under a Windows 10 operating system on a computer with 32 GB RAM and a 2.90-GHz Intel Core i7-10700 CPU.

B. Parameter Settings

1) *Parameter Settings of the Test Problems*: For each test problem, the change frequency (τ_t) was set to 10, 14, and 18, and the number (n_t) of dynamic environments was 21 (i.e., $t \in [0,1,\ldots,20]$). Each algorithm was run for $40+\tau_t$ generations in the first environment so that the effect of static optimization could be minimized. The total number of iterations was $n_t \times \tau_t+40$. Each test problem includes ten decision variables: $x_1 \in [0,1]$, and $x_j \in [0,2]$ for $j \in [2, 10]$. The parameters in $g(\mathbf{x_{II}}, t)$ and $g(\mathbf{x_{III}}, t)$ are set as follows: jl=2, ju=4, c=2, η=2.1, tl=5, λ=0.5, α=0.1, β=0.52136, k=0.05, ε=6, and χ=0.38. The other parameters of the test problems are determined according to [2]. The optimal value of a distance variable that changes more than 0.1 is considered to have a drastic change in a dynamic environment. Note that these parameters can also be changed according to specific requirements of problem characteristics. The approximate feasibility ratios of the test problems at each environment are given in Section S-V of the Supplementary Material. In that section, this paper also gives the true DPOFs and DPOSs of eight problems.

2) *Algorithm Parameters*: The population size was set to 200 for both parent and offspring populations. According to our previous study in [2], the crossover probability and the distribution index were set to 0.8 and 5 in SBX, respectively. The mutation probability and the distribution index were set to 0.05 and 40, respectively, in PM. The effects of the above parameters are given in Section S-VI of the Supplementary Material. The other parameters for the six compared algorithms were the same as those used in the referenced papers.

C. Performance Indicators

We adopt two performance metrics, i.e., the hypervolume (HV) [89] and inverted generational distance (IGD) [90], to evaluate all competing algorithms at the end of each change period, which are formulated in Equations (11) and (12), respectively.

$$HV\left(DPOF_t^*\right) = VOL\left(\bigcup_{\mathbf{x}\in A}[f_1(\mathbf{x}), z_1]\times\cdots\times[f_m(\mathbf{x}), z_m]\right) \tag{11}$$

$$IGD\left(DPOF_t, DPOF_t^*\right) = \frac{1}{|DPOF_t|}\sum_{i=1}^{|DPOF_t|} d_i \tag{12}$$

where $DPOF_t$ and $DPOF_t^*$ represent a uniformly distributed true DPOF and the DPOF obtained by the underlying algorithm at time t, respectively. $VOL(\cdot)$ denotes the Lebesgue measure. The reference points $(z_1,\ldots,z_m)$ are set as $(z_1^*+1,\ldots z_m^*+1)$, and z_m^* is the maximum value of the true DPOFs on the mth objective. IGD calculates the average Euclidean distance from the reference true DPOF to DPOF* at time t. d_i is the minimum Euclidean distance between the ith individual in DPOF and individuals in DPOF*.

Each test problem includes n_t dynamic environments. Therefore, this paper uses the mean HV (MHV) and mean IGD (MIGD) [94] values to compare all algorithms, which are as follows:

TABLE II
PERFORMANCE RANKING ON TWO METRICS FOR TEST PROBLEMS.

Instance No.	Rank	Ranking by MHV	Ranking by MIGD	Instance No.	Rank	Ranking by MHV	Ranking by MIGD
1	1th	*m*EDCMOA	*m*EDCMOA	5	1th	*m*EDCMOA	*m*EDCMOA
	2nd	DC-NSGA-II	DC-NSGA-II		2nd	DC-NSGA-II	DC-NSGA-II
	3rd	*d*CMOEA	*d*CMOEA		3rd	*d*CMOEA	*d*CMOEA
	4th	DC-NSGA-III	DC-NSGA-III		4th	DC-NSGA-III	DC-NSGA-III
	5th	DC-NSGA-II-A	DC-NSGA-II-A		5th	DC-NSGA-II-A	DC-NSGA-II-A
	6th	DC-TAEA	DC-TAEA		6th	DC-TAEA	DC-TAEA
	7th	DC-MOEA	DC-MOEA		7th	DC-MOEA	DC-MOEA
2	1th	*m*EDCMOA	*m*EDCMOA	6	1th	*m*EDCMOA	*m*EDCMOA
	2nd	DC-NSGA-II	DC-NSGA-II		2nd	*d*CMOEA	DC-NSGA-II
	3rd	*d*CMOEA	*d*CMOEA		3rd	DC-NSGA-II	*d*CMOEA
	4th	DC-NSGA-III	DC-NSGA-III		4th	DC-NSGA-III	DC-NSGA-III
	5th	DC-NSGA-II-A	DC-NSGA-II-A/ DC-TAEA		5th	DC-NSGA-II-A	DC-NSGA-II-A
	6th	DC-TAEA	DC-MOEA		6th	DC-TAEA	DC-TAEA
	7th	DC-MOEA			7th	DC-MOEA	DC-MOEA
3	1th	*m*EDCMOA	*m*EDCMOA	7	1th	*m*EDCMOA	*m*EDCMOA
	2nd	DC-NSGA-II/ *d*CMOEA	DC-NSGA-II		2nd	*d*CMOEA	DC-NSGA-II
	3rd	DC-NSGA-III	*d*CMOEA		3rd	DC-NSGA-II	*d*CMOEA
	4th	DC-NSGA-II-A	DC-NSGA-III		4th	DC-NSGA-III	DC-NSGA-III
	5th	DC-TAEA	DC-TAEA		5th	DC-NSGA-II-A	DC-TAEA
	6th	DC-MOEA	DC-NSGA-II-A		6th	DC-TAEA	DC-NSGA-II-A
	7th		DC-MOEA		7th	DC-MOEA	DC-MOEA
4	1th	*m*EDCMOA	*m*EDCMOA	8	1th	*m*EDCMOA	*m*EDCMOA
	2nd	DC-NSGA-II	DC-NSGA-II		2nd	DC-NSGA-II	DC-NSGA-II
	3rd	*d*CMOEA	*d*CMOEA		3rd	*d*CMOEA	*d*CMOEA
	4th	DC-NSGA-III	DC-NSGA-III		4th	DC-NSGA-III	DC-NSGA-III
	5th	DC-NSGA-II-A	DC-TAEA		5th	DC-NSGA-II-A	DC-NSGA-II-A
	6th	DC-TAEA	DC-NSGA-II-A		6th	DC-TAEA	DC-TAEA
	7th	DC-MOEA	DC-MOEA		7th	DC-MOEA	DC-MOEA

$$MHV = \frac{1}{n_t}\sum_{t=1}^{n_t} HV\left(DPOF_t^*\right) \tag{13}$$

$$MIGD = \frac{1}{n_t}\sum_{t=1}^{n_t} IGD\left(DPOF_t, DPOF_t^*\right) \tag{14}$$

The higher MHV value and the smaller MIGD value indicate that the obtained solutions in DPOS* have better quality.

VI. EXPERIMENTAL RESULTS

In this section, the effectiveness of the improved test problems is first examined in Section VI-A. Subsequently, the performance of *m*EDCMOA in comparison to other algorithms is explored in Section VI-B. Then, we verify the contributions of the proposed components of *m*EDCMOA, followed by the effectiveness of *m*EDCMOA on the test problems with ju=6 and 8. Finally, this paper uses other indicators to further verify the effectiveness of *m*EDCMOA. For each combination (n_t, τ_t) for a test problem, each algorithm was run 30 times, and the mean and standard deviations of the MHV and MIGD values are given in the corresponding tables. This paper exploits the Wilcoxon rank-sum test at the 0.05 significance level to determine whether one algorithm statistically differs from others concerning the performance metrics.

A. Effectiveness of Test Problems

This paper exploits the ranking method described in [91] to verify the effectiveness of the designed test problems with a generator for diversifying the optimal values of the distance variables. Specifically, the average ranking of each algorithm on three combinations was first calculated. Second, an algorithm is given the highest ranking if it surpasses other algorithms with the largest number; conversely, this algorithm is given the worst ranking if it does not convincingly defeat other algorithms. If multiple algorithms outperform the same number of others, then they have the same ranking. Finally, the rankings of the seven algorithms are listed in Table II.

The findings show that *m*EDCMOA achieves the best

TABLE III
MEAN AND STANDARD DEVIATIONS OF MHV VALUES OBTAINED BY SEVEN ALGORITHMS

Ins	(τ_t, n_t)	*d*CMOEA	DC-MOEA	DC-NSGA-II-A	DC-NSGA-II	DC-NSGA-III	DC-TAEA	*m*EDCMOA
1	(10, 21)	3.29111(0.02918)‡	2.60788(0.07958)‡	2.99069(0.06355)‡	3.33632(0.02146)‡	3.19375(0.02939)‡	2.97091(0.07045)‡	**3.51923(0.00224)**
	(14, 21)	3.42074(0.01587)‡	3.13119(0.05371)‡	3.23297(0.05743)‡	3.42717(0.01655)‡	3.33111(0.01207)‡	3.18398(0.06479)‡	**3.52701(0.00146)**
	(18, 21)	3.46955(0.01522)‡	3.32198(0.02964)‡	3.34898(0.04646)‡	3.47159(0.00898)‡	3.41383(0.01254)‡	3.30525(0.05150)‡	**3.53070(0.00143)**
2	(10, 21)	3.23453(0.02350)‡	2.71190(0.10551)‡	2.94648(0.05731)‡	3.26872(0.01754)‡	3.13222(0.02725)‡	3.00862(0.07031)‡	**3.41524(0.00163)**
	(14, 21)	3.34191(0.00934)‡	3.10859(0.03562)‡	3.20344(0.04256)‡	3.35145(0.00666)‡	3.26100(0.01759)‡	3.17427(0.06025)‡	**3.42030(0.00121)**
	(18, 21)	3.38324(0.00528)‡	3.27521(0.02498)‡	3.31403(0.02554)‡	3.38123(0.00535)‡	3.33346(0.00968)‡	3.29424(0.03678)‡	**3.42259(0.00183)**
3	(10, 21)	3.31082(0.02386)‡	2.87014(0.11405)‡	2.96434(0.06203)‡	3.34270(0.02110)‡	3.19653(0.02162)‡	3.02157(0.07721)‡	**3.51969(0.00244)**
	(14, 21)	3.43271(0.01016)‡	3.21572(0.05325)‡	3.25339(0.05628)‡	3.43233(0.01428)‡	3.34002(0.01756)‡	3.24835(0.07004)‡	**3.52703(0.00209)**
	(18, 21)	3.48086(0.00505)‡	3.38107(0.03086)‡	3.38920(0.02830)‡	3.47560(0.00949)‡	3.41589(0.01139)‡	3.38631(0.03477)‡	**3.53175(0.00179)**
4	(10, 21)	3.20315(0.03319)‡	2.82626(0.06753)‡	2.93973(0.05149)‡	3.25633(0.02084)‡	3.12189(0.02977)‡	3.00224(0.05765)‡	**3.40792(0.00120)**
	(14, 21)	3.31520(0.01729)‡	3.12333(0.04496)‡	3.20029(0.05143)‡	3.33587(0.00862)‡	3.25801(0.01541)‡	3.16743(0.05528)‡	**3.41214(0.00133)**
	(18, 21)	3.36551(0.00984)‡	3.28336(0.01844)‡	3.30380(0.03670)‡	3.36991(0.00832)‡	3.32092(0.00948)‡	3.29406(0.02412)‡	**3.41503(0.00094)**
5	(10, 21)	3.30338(0.03464)‡	2.64552(0.08790)‡	3.01041(0.07356)‡	3.34900(0.02044)‡	3.20276(0.02598)‡	3.00976(0.08375)‡	**3.53629(0.00206)**
	(14, 21)	3.44756(0.01503)‡	3.14793(0.05598)‡	3.24962(0.05383)‡	3.44201(0.01547)‡	3.35482(0.02003)‡	3.19756(0.05644)‡	**3.54228(0.00151)**
	(18, 21)	3.49307(0.00938)‡	3.35269(0.04002)‡	3.39109(0.04038)‡	3.49033(0.00860)‡	3.42651(0.01378)‡	3.30477(0.06743)‡	**3.54773(0.00180)**
6	(10, 21)	3.26086(0.02448)‡	2.73378(0.08300)‡	2.96681(0.05417)‡	3.29191(0.01072)‡	3.16512(0.02023)‡	3.02341(0.08639)‡	**3.44291(0.00181)**
	(14, 21)	3.37331(0.00747)‡	3.12063(0.04375)‡	3.24821(0.03909)‡	3.37244(0.01091)‡	3.29519(0.01722)‡	3.18518(0.06512)‡	**3.44843(0.00129)**
	(18, 21)	3.40892(0.00610)‡	3.29977(0.01569)‡	3.33753(0.03716)‡	3.40521(0.00524)‡	3.35703(0.00830)‡	3.30618(0.04781)‡	**3.45112(0.00139)**
7	(10, 21)	3.33332(0.02107)‡	2.83521(0.08562)‡	2.96270(0.06160)‡	3.35342(0.01833)‡	3.21681(0.03276)‡	3.05683(0.07316)‡	**3.53560(0.00187)**
	(14, 21)	3.44812(0.01363)‡	3.23180(0.05815)‡	3.26908(0.05818)‡	3.44212(0.01825)‡	3.35761(0.01853)‡	3.27788(0.06205)‡	**3.54243(0.00226)**
	(18, 21)	3.50020(0.00517)‡	3.38714(0.03444)‡	3.40275(0.04566)‡	3.48951(0.01038)‡	3.42786(0.01568)‡	3.38476(0.04462)‡	**3.54703(0.00161)**
8	(10, 21)	3.23809(0.02732)‡	2.81100(0.09003)‡	2.94779(0.06063)‡	3.28129(0.02333)‡	3.15673(0.02452)‡	3.01202(0.05968)‡	**3.43758(0.00156)**
	(14, 21)	3.35015(0.01688)‡	3.15394(0.04858)‡	3.24521(0.03863)‡	3.36642(0.01015)‡	3.28533(0.01371)‡	3.19473(0.06264)‡	**3.44294(0.00124)**
	(18, 21)	3.40164(0.00688)‡	3.29951(0.02237)‡	3.33119(0.03239)‡	3.40066(0.00829)‡	3.35257(0.00860)‡	3.30823(0.04627)‡	**3.44574(0.00103)**

‡ and † indicate *d*CMOEA performs significantly better than and equivalently to the corresponding algorithm, respectively.

TABLE IV
MEAN AND STANDARD DEVIATIONS OF MIGD VALUES OBTAINED BY SEVEN ALGORITHMS

Ins	(τ_t, n_t)	*d*CMOEA	DC-MOEA	DC-NSGA-II-A	DC-NSGA-II	DC-NSGA-III	DC-TAEA	*m*EDCMOA
1	(10, 21)	0.08736(0.00933)‡	0.34018(0.03545)‡	0.19137(0.03326)‡	0.06470(0.00559)‡	0.10400(0.00687)‡	0.18130(0.04867)‡	**0.01169(0.00090)**
	(14, 21)	0.04330(0.00512)‡	0.14497(0.02175)‡	0.10872(0.02729)‡	0.03492(0.00347)‡	0.06085(0.00331)‡	0.11070(0.03366)‡	**0.00945(0.00088)**
	(18, 21)	0.02582(0.00427)‡	0.07660(0.00960)‡	0.06547(0.02039)‡	0.02226(0.00164)‡	0.03673(0.00235)‡	0.08470(0.02660)‡	**0.00840(0.00088)**
2	(10, 21)	0.07926(0.00862)‡	0.27283(0.04628)‡	0.18425(0.03074)‡	0.05767(0.00544)‡	0.09810(0.00797)‡	0.13931(0.03113)‡	**0.01135(0.00094)**
	(14, 21)	0.03853(0.00361)‡	0.12400(0.01433)‡	0.08754(0.01910)‡	0.03019(0.00202)‡	0.05586(0.00409)‡	0.08777(0.02508)‡	**0.01046(0.00104)**
	(18, 21)	0.02202(0.00233)‡	0.06230(0.00843)‡	0.04392(0.00861)‡	0.02023(0.00130)‡	0.03243(0.00250)‡	0.04752(0.01356)‡	**0.00936(0.00124)**
3	(10, 21)	0.08237(0.00857)‡	0.28254(0.04069)‡	0.19431(0.02568)‡	0.06436(0.00589)‡	0.10479(0.00492)‡	0.16191(0.03129)‡	**0.01236(0.00103)**
	(14, 21)	0.04069(0.00412)‡	0.14963(0.01724)‡	0.09934(0.02110)‡	0.03492(0.00240)‡	0.06039(0.00450)‡	0.09407(0.03151)‡	**0.01075(0.00128)**
	(18, 21)	0.02324(0.00147)‡	0.08128(0.00865)‡	0.04960(0.00691)‡	0.02223(0.00128)‡	0.03670(0.00307)‡	0.04958(0.01224)‡	**0.00949(0.00091)**
4	(10, 21)	0.08596(0.01151)‡	0.22697(0.03096)‡	0.18637(0.02844)‡	0.05838(0.00526)‡	0.09693(0.00803)‡	0.14658(0.03517)‡	**0.01033(0.00085)**
	(14, 21)	0.04493(0.00624)‡	0.11853(0.01662)‡	0.08707(0.02973)‡	0.03109(0.00261)‡	0.05268(0.00393)‡	0.08926(0.02688)‡	**0.00941(0.00093)**
	(18, 21)	0.02558(0.00369)‡	0.05800(0.00629)‡	0.04556(0.01423)‡	0.01966(0.00132)‡	0.03212(0.00173)‡	0.04341(0.00748)‡	**0.00861(0.00118)**
5	(10, 21)	0.08990(0.01104)‡	0.33212(0.04339)‡	0.19445(0.04064)‡	0.07046(0.00513)‡	0.10619(0.00790)‡	0.17935(0.04551)‡	**0.02616(0.00122)**
	(14, 21)	0.04503(0.00420)‡	0.14677(0.01856)‡	0.10950(0.02577)‡	0.04469(0.00374)‡	0.06274(0.00426)‡	0.11943(0.03440)‡	**0.02407(0.00090)**
	(18, 21)	0.03315(0.00285)‡	0.07875(0.01259)‡	0.05822(0.01049)‡	0.03238(0.00147)‡	0.04351(0.00293)‡	0.09819(0.03647)‡	**0.02339(0.00083)**
6	(10, 21)	0.07913(0.00800)‡	0.27101(0.03602)‡	0.17660(0.03355)‡	0.06204(0.00386)‡	0.09428(0.00534)‡	0.14508(0.03960)‡	**0.02611(0.00114)**
	(14, 21)	0.04157(0.00245)‡	0.12692(0.01560)‡	0.07814(0.01322)‡	0.03707(0.00196)‡	0.05589(0.00503)‡	0.09434(0.02601)‡	**0.02459(0.00106)**
	(18, 21)	0.03042(0.00182)‡	0.06423(0.00516)‡	0.05032(0.01402)‡	0.02912(0.00097)‡	0.03871(0.00240)‡	0.05647(0.01750)‡	**0.02387(0.00078)**
7	(10, 21)	0.08319(0.00744)‡	0.29051(0.03934)‡	0.20380(0.02599)‡	0.06939(0.00452)‡	0.10258(0.00816)‡	0.15164(0.03081)‡	**0.02637(0.00090)**
	(14, 21)	0.04576(0.00378)‡	0.14564(0.01941)‡	0.10172(0.02382)‡	0.04386(0.00372)‡	0.06372(0.00499)‡	0.08971(0.02722)‡	**0.02490(0.00091)**
	(18, 21)	0.03152(0.00133)‡	0.08512(0.01080)‡	0.05829(0.01625)‡	0.03317(0.00229)‡	0.04410(0.00322)‡	0.05961(0.01666)‡	**0.02402(0.00091)**
8	(10, 21)	0.08319(0.01020)‡	0.25119(0.04285)‡	0.19338(0.04426)‡	0.06229(0.00626)‡	0.09429(0.00742)‡	0.14845(0.03625)‡	**0.02471(0.00112)**
	(14, 21)	0.04679(0.00633)‡	0.11485(0.01695)‡	0.07861(0.01811)‡	0.03784(0.00239)‡	0.05644(0.00463)‡	0.09389(0.03840)‡	**0.02363(0.00070)**
	(18, 21)	0.03035(0.00222)‡	0.06546(0.00813)‡	0.05037(0.01164)‡	0.02957(0.00127)‡	0.03831(0.00207)‡	0.05504(0.01885)‡	**0.02310(0.00085)**

ranking in terms of MHV and MIGD metrics, followed by *d*CMOEA or DC-NSGA-II, which might be attributed to using a similar method for selecting parents and a similar population update strategy for updating the population in *m*EDCMOA, *d*CMOEA and DC-NSGA-II. DC-MOEA has the worst performance among these seven algorithms. Considering the MHV ranking, DC-NSGA-II-A, DC-NSGA-III, and DC-TAEA perform at different ranks. The MIGD value ranking reveals that DC-NSGA-II-A and DC-TAEA show considerable performance on the second test problem, and DC-NSGA-II-A, DC-NSGA-III, and DC-TAEA have different ranks on other instances. The above results mean that the test problems are effective at distinguishing different algorithms.

B. Algorithm Comparison

This section compares *m*EDCMOA with six algorithms (i.e., *d*CMOEA, DC-MOEA, DC-NSGA-II-A, DC-NSGA-II, DC-NSGA-III, and DC-TAEA). Tables III and IV present comparison results of the above seven algorithms in terms of MHV and MIGD values, respectively.

From these comparison results of the MHV and MIGD values, it is clear that the proposed *m*EDCMOA has an excellent performance in solving these eight benchmark problems with different change frequencies, which is closely followed by DC-NSGA-II, *d*CMOEA, and DC-NSGA-III. DC-MOEA performs the worst among the seven algorithms. From the results of the MHV metric, *d*CMOEA and DC-NSGA-II have similar performance on the first, third, fifth, sixth, and seventh test problems with τ_t=14 and the first, second, fourth, fifth, sixth, and eighth test problems with τ_t=18. DC-MOEA and DC-TAEA perform similarly on the third, sixth, seventh, and eighth test problems with τ_t=18, and DC-MOEA is better than DC-TAEA in other instances. From the results of the MIGD metric, *d*CMOEA and DC-NSGA-II perform comparably on the third, fifth, sixth, and eighth test problems with τ_t=18. This might be because they have the same penalty method for infeasible solutions. DC-NSGA-II-A and

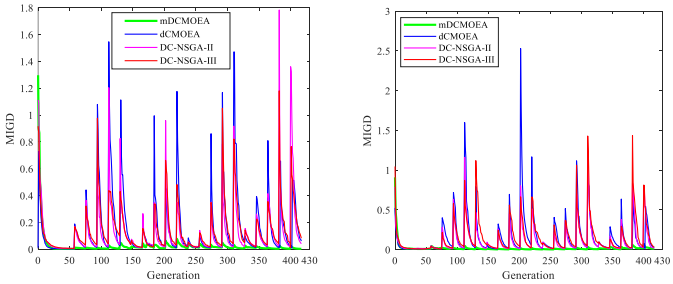

Fig. 2. Evolution curves of average MIGD values for the first (left) and second (right) instances with τ_t=18 and n_t=21.

DC-TAEA have similar MIGD values for the first, second, and fourth test problems with τ_t=14 and the third, fourth, and seventh test problems with τ_t=18. Overall, the experimental results on the MHV and MIGD metrics demonstrate the optimization capability of *m*EDCMOA in solving DCMOPs with diverse changes in the optimal distance variables. In fact, *m*EDCMOA and *d*CMOEA borrow from DC-NSGA-II the idea of setting a penalty for infeasible solutions, and the relatively poor performance of *d*CMOEA and DC-NSGA-II may be due to the disadvantages of their dynamic response method and population update strategies.

As a supplement to the tabular presentation, Fig. 2 gives the evolutionary curves of the IGD values on the first and second test problems with τ_t=18 and n_t=21, and the curves on the other test problems are shown in Section S-VII of the Supplementary Material. Note that this paper does not show the evolutionary curves of DC-MOEA, DC-NSGA-II-A, and DC-TAEA on the MIGD metrics due to their poor performance. As shown in Fig. 2 and the figures in Section S-VII of the Supplementary Material, the proposed *m*EDCMOA can respond to diverse dynamic environments more steadily and converge faster for test problems, meaning that it performs significantly better than the other compared algorithms. Additionally, these results show that the proposed *m*EDCMOA has a good convergence capability.

To further verify the tracking capability of *m*EDCMOA, this paper plots the final DPOFs of *m*EDCMOA, *d*CMOEA, DC-NSGA-II, and DC-NSGA-III on the first test problem with τ_t=18 and n_t=21, which is shown in Fig. 3, and the final DPOFs on other test problems can be seen in Section S-VIII of the Supplementary Material. The tracking figures validate that *m*EDCMOA has excellent tracking capability compared with other algorithms available in the literature.

C. Study of Different Components of mEDCMOA

The proposed *m*EDCMOA includes three main components, i.e., the tribe classification operator, the population update operator, and the dynamic response strategy. To validate the effectiveness of each component, we exploit different strategies published in the literature to replace the corresponding approaches of *m*EDCMOA, forming different variants. Specifically, the tribe classification operator of *m*EDCMOA is used to select nondominated infeasible solutions, thus we replace it with the nondominated infeasible solution selection method proposed in [33]. This variant is called *m*EDCMOA-V1. *m*EDCMOA-V2 is a variant of *m*EDCMOA that uses the constraint handling strategy in [4]. The third and fourth

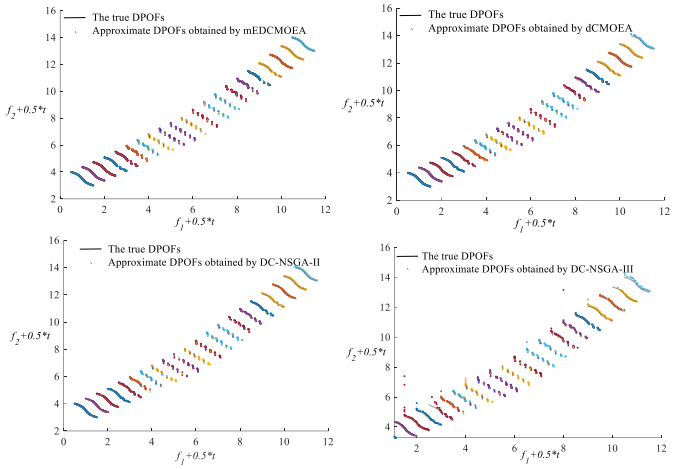

Fig. 3. Obtained DPOFs obtained by *m*EDCMOA (top left), *d*CMOEA (top right), DC-NSGA-II (bottom left), and DC-NSGA-III (bottom right) on the first test problem with τ_t=18 and n_t=21.

variants (i.e., *m*EDCMOA-V3 and *m*EDCMOA-V4) adopt the population selection strategy designed in [2] and in [4], respectively, to replace the population update method of *m*EDCMOA. *m*EDCMOA-V5 and *m*EDCMOA-V6 are the fifth and sixth variants, respectively, in which the dynamic response strategy of *m*EDCMOA is replaced by those proposed in [2] and [4]. The above variants were compared with *m*EDCMOA on the test problems with settings of (τ_t, n_t)=(10, 21), (14, 21), and (18, 21). The average values and standard deviations of MHV and MIGD metrics and discussions are given in Section S-IX of the Supplementary Material.

In summary, the key components designed in *m*EDCMOA play critical roles in solving DCMOPs efficiently and effectively. Hence, this paper explains the unique role of each component in improving the algorithm performance in detail. Specifically, the tribe classification operator divides solutions into feasible solutions, and dominated and nondominated infeasible solutions, encouraging solutions with different performances to search for DPOSs under diverse environments. The population update operator can retain nondominated solutions when the number of feasible solutions is insufficient, pushing the population toward the DPOFs and reserving nondominated infeasible solutions with promising behaviors. The dynamic response strategy can update solutions according to the movement directions and magnitudes of the distance variables in different tribes, accelerating the tracking speed of the population for the DPOFs.

D. Influence of the Number of Optimal Distance Variables with Drastic Changes

The above experimental results are obtained when *ju* is set to 4. To verify that the proposed algorithm can effectively solve DCMOPs with different numbers of optimal distance variables that have drastic changes in a dynamic environment, *ju* is revised to 6 and 8. Considering that DC-MOEA, DC-NSGA-II-A, and DC-TAEA perform poorly on the test problem with *ju*=4, the test problems with *ju*=6 and *ju*=8 are solved using only *m*EDCMOA, *d*CMOEA, DC-NSGA-II, and DC-NSGA-III. The corresponding results and discussion are given in Section S-X of the Supplementary Material.

E. Experimental Results on Other Performance Measures

The offline error and feasible time ratio have been used for evaluating the capability of an algorithm to deal with the dynamic changes of the single-objective optimization problems [50], [85]. The offline error is calculated according to the best solution in each generation. However, unlike single-objective optimization problems, a set of solutions can satisfy the optima of a given DCMOP. Therefore, the offline error designed for single-objective optimization problems cannot be directly used for evaluating the Pareto-optimal solutions of DCMOPs. This paper modifies the offline error based on the IGD metric so that it can be used for evaluating Pareto-optimal solutions. The corresponding discussion and results are given in Section S-XI of the Supplementary Material.

VII. Conclusion

This paper proposes an *m*EDCMOA to solve DCMOPs with diverse changes in dynamic objectives, constraints, and distance variables. The tribe classification operator of *m*EDCMOA classifies the population into different tribes and can effectively identify potential infeasible solutions. To facilitate nondominated solutions to drive other solutions with large objective values toward the DPOFs, we design a population selection strategy to retain nondominated infeasible solutions with promising objective values when feasible solutions are not enough. When a changed environment arises, the dynamic response strategy evaluates the direction and magnitude of each distance variable moving toward the DPOSs and updates other solutions in different tribes, obtaining a reinitialized population close to the new DPOFs. Additionally, we design a generator to ensure that the optimal values of the distance variables have different changes when dynamic environments vary.

Although that *m*EDCMOA has shown promising performance, some interesting topics on DCMOPs are worthy of further study. First, although *m*EDCMOA performs well on the test problems with two objectives, it may be defeated by other algorithms in optimizing the problems with three or more objectives. Thus, a strategy that can handle many-objective optimization problems should be exploited in the future. Second, the one-dimensional search strategy used in the dynamic response strategy does not consider the problem variable interaction structure, which may cause low response efficiency in solving other DCMOPs. Thus, the dynamic response strategy that considers the problem variable interaction structure should be designed. Finally, additional costs incurred by switching one solution to another [92] and a robust framework [93] could be considered.

References

[1] Q. Chen, J. Ding, T. Chai, and Q. Pan, "Evolutionary optimization under uncertainty: the strategies to handle varied constraints for fluid catalytic cracking operation," *IEEE Trans. Cyber.*, vol. 52, no. 4, pp. 2249–2262, Apr. 2022.

[2] Q. Chen, J. Ding, S. Yang, and T. Chai, "A novel evolutionary algorithm for dynamic constrained multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 792–806, Aug. 2020.

[3] W. Kong, T. Chai, S. Yang, and J. Ding, "A hybrid evolutionary multiobjective optimization strategy for the dynamic power supply problem in magnesia grain manufacturing," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2960-2969, May 2013.

[4] R. Azzouz, S. Bechikh, L. B. Said, and W. Trabelsi, "Handing time-varying constraints and objectives in dynamic evolutionary multiobjective optimization," *Swarm Evol. Comput.*, vol. 39, pp. 222–248, Apr. 2018.

[5] R. Azzouz, S. Bechikh, and L. Ben Said, "Multiobjective optimization with dynamic constraints and objective: new challenges for evolutionary algorithm," in *Proc. Annu. Conf. Genet. Evol. Comput*. (*GECCO*), 2016, pp. 615–622.

[6] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.

[7] X. Lu, K, Tang, S, Menzel, and X. Yao, "Dynamic optimization in fast-changing environments via offline evolutionary search," *IEEE Trans. Evol. Comput.*, vol. 26, no. 3, pp. 431–445, Jun. 2022.

[8] J. Liang, X. Ban, K. Yu, B. Qu, K. Qiao, C. Yue, K. Chen, and K. C. Tan, "A survey on evolutionary constrained multi-objective optimization," *IEEE Trans. Evol. Comput.*, 2022, DOI: 10.1109/TEVC.2022.3155533.

[9] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, "Constraint handling in multiobjective evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 514–525, Jun. 2009.

[10] H. M. Maldonado and S. Zapotecas-Martinez, "A dynamic penalty function within MOEA/D for constrained multi-objective optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 2021, pp. 1470–1477.

[11] K. Yu, J. Liang, B. Qu, Y. Luo, and C. Yue, "Dynamic selection preference-assisted constrained multiobjective differential evolution," *IEEE Trans. Syst., Man, Cybern., Syst*., vol. 52, no. 5, pp.2954–2965, Mar. 2022.

[12] Z. Ma and Y. Wang, "Shift-based penalty for evolutionary constrained multiobjective optimization and its application," *IEEE Trans. Cyber*., pp. 1–13, 2021, DOI: 10.1109/TCYB.2021.3069814.

[13] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Aug. 2002.

[14] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *Proc. IEEE Congr. Evol. Comput.*, 2006, pp. 1–8.

[15] C. A. Coello Coello and A. D. Christiansen, "MOSES: A multiobjective optimization tool for engineering design," *Eng. Opt.*, vol. 31, no. 3, pp. 337–368, 1999.

[16] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2014.

[17] Z. Fan, W. Li, X. Cai, K. Hu, H. Lin, and H. Li, "Angle-based constrained dominance principle in MOEA/D for constrained multiobjective optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 460-467.

[18] Z. Fan, Y. Fang, W. Li, X. Cai, C. Wei, and E. Goodman, "MOEA/D with angle-based constrained dominance principle for constrained multi-objective optimization problems," *Appl. Soft Comput.*, vol. 74, pp. 621–633, Jan. 2019.

[19] D. K. Saxena, T. Ray, K. Deb, and A. Tiwari, "Constrained many-objective optimization: A way forward," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 545–552.

[20] S. Z Martinez, A. Montano, and C. A. C Coello, "Constrained multi-objective aerodynamic shape optimization via swarm intelligence," in *Proc. Annu. Conf. Genet. Evol. Comput*. (*GECCO*), 2014, pp. 81–88.

[21] Q. Zhu, Q. Zhang, and Q. Lin, "A constrained multiobjective evolutionary algorithm with detect-and-escape strategy," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, pp. 938–947, Mar. 2020.

[22] M. A. Jan and R. A. Khanum, "A study of two penalty-parameterless constraint handling techniques in the framework of MOEA/D," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 128–148, Jan. 2013.

[23] Z.-Z. Liu, Y. Wang, and B.-C. Wang, "Indicator-based constrained multiobjective evolutionary algorithms," *IEEE Trans. Syst., Man, Cybern., Syst*., vol. 51, no. 9, pp. 5414–5426, Sept. 2021.

[24] D. A. Vieira, R. L. Adriano, J. A. Vasconcelos, and L. Krahenbuhl, "Treating constraints as objectives in multiobjective optimization problems using niched Pareto genetic algorithm," *IEEE Trans. Magn.*, vol. 40, no. 2, pp. 1188–1191, Mar. 2004.

[25] Q. Long, "A constraint handling technique for constrained multi-objective genetic algorithm," *Swarm Evol. Comput.*, vol. 15, pp. 66–79, Apr. 2014.

[26] C. Peng, H. Liu, and F. Gu, "An evolutionary algorithm with directed weights for constrained multiobjective optimization," *Appl. Soft Comput.*,

vol. 60, pp. 613–622, Nov. 2017.
[27] Y. Zhou, M. Zhu, J. Wang, Z. Zhang, Y. Xiang, and J. Zhang, "Tri-goal evolution framework for constrained many-objective optimization," *IEEE Trans. Syst., Man, Cybern., Syst*., vol. 50, no. 8, pp. 3086–3099, Aug. 2020.
[28] J. Wang, G. Liang, and J. Zhang, "Cooperative differential evolution framework for constrained multiobjective optimization," *IEEE Trans. Cyber*., vol. 49, no. 6, pp. 2060–2072, Jun. 2019.
[29] Y. Yang, J. Liu, and S. Tan, "A partition-based constrained multi-objective evolutionary algorithm," *Swarm Evol. Comput.*, vol. 66, pp. 100940, Oct. 2021.
[30] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin, "A coevolutionary framework for constrained multi-objective optimization problems," *IEEE Trans. Evol. Comput*., vol. 25, no. 1, pp. 102–116, Feb. 2021.
[31] J. Wang, Y. Li, Q. Zhang, Z. Zhang, and S. Gao, "Cooperative multiobjective evolutionary algorithm with propulsive population for constrained multiobjective optimization," *IEEE Trans. Syst., Man, Cybern., Syst*., vol. 52, no. 6, pp. 3476–3491, Jun. 2022.
[32] K. Li, R. Chen, G. Fu, and X. Yao, "Two-archive evolutionary algorithm for constrained multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 303–315, Apr. 2019.
[33] Z.-Z. Liu, B.-C. Wang, and K. Tang, "Handling constrained multiobjective optimization problems via bidirectional coevolution," *IEEE Trans. Cyber*., 2021, vol. 52, no. 10, pp. 10163–10176, Oct. 2022.
[34] L. V. Santana-Quintero, A. G. Hernandez-Dıaz, J. Molina, C. A. Coello. Coello, and R. Caballero, "DEMORS: A hybrid multi-objective optimization algorithm using differential evolution and rough set theory for constrained problems," *Comput. Oper. Res*., vol. 37, no. 3, pp. 470–480, Mar. 2010.
[35] Z. Fan, Z. Wang, W. Li, Y. Yuan, Y. You, Z. Yang, F. Sun, and J. Ruan, "Push and pull search embedded in an M2M framework for solving constrained multiobjective optimization problems," *Swarm Evol. Comput.*, vol. 54, pp. 100651, May 2020.
[36] F. Ming, W. Gong, H. Zhen, S. Li, L. Wang, and Z. Liao, "A simple two-stage evolutionary algorithm for constrained multi-objective optimization," *Knowl-Based Syst*., vol. 228, no. 27, pp. 107263, Sept. 2021.
[37] Y. Tian, Y. Zhang, Y. Su, X. Zhang, K. C. Tan, and Y. Jin, "Balancing objective optimization and constraint satisfaction in constrained evolutionary multiobjective optimization," *IEEE Trans. Cyber*., vol. 52, no. 9, pp. 9559–9572, Sep. 2022.
[38] Z.-Z. Liu and Y. Wang, "Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 870–884, Oct. 2019.
[39] K. Yu, J. Liang, B. Qu, and C. Yue, "Purpose-directed two-phase multiobjective differential evolution for constrained multiobjective optimization," *Swarm Evol. Comput.*, vol. 60, pp. 100799, Feb. 2021.
[40] J Liang, K Qiao, K Yu, B Qu, C Yue, W Guo, L Wang, "Utilizing the relationship between unconstrained and constrained Pareto fronts for constrained multiobjective optimization," *IEEE Trans. Cyber*., 2022, DOI: 10.1109/TCYB.2022.3163759.
[41] V. Kelner, F. Capitanescu, O. Leonard, and L. Wehenkel, "A hybrid optimization technique coupling an evolutionary and a local search algorithm," *J. Comput. Appl. Math*., vol. 215, no. 2, pp. 448–456, Jun. 2008.
[42] S. Datta, A. Ghosh, K. Sanyal, and S. Das, "A radial boundary intersection aided interior point method for multi-objective optimization," *Inf. Sci.*, vol. 377, pp. 1–16, Jan. 2017.
[43] L. Uribe, A. Lara, and O. Schutze, "On the efficient computation and use of multi-objective descent directions within constrained MOEAs," *Swarm Evol. Comput.*, vol. 52, pp. 100617, Feb. 2020.
[44] V. A. S. Hernandez, O. Schutze, H. Wang, A. Deutz, and M. Emmerich, "The set-based hypervolume newton method for bi-objective optimization," *IEEE Trans. Cyber*, vol. 50, no. 5, pp. 2186–2196, May 2020.
[45] X. Yu, X. Yu, Y. Lu, G. G. Yen, and M. Cai, "Differential evolution mutation operators for constrained multi-objective optimization," *Appl. Soft Comput.*, vol. 67, pp. 452–466, Jun. 2018.
[46] C. He, R. Cheng, Y. Tian, X. Zhang, K. C. Tan, and Y. Jin, "Paired offspring generation for constrained large-scale multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 448–462, Jun. 2021.
[47] R. Datta and R. G. Regis, "A surrogate-assisted evolution strategy for constrained multi-objective optimization," *Expert Syst. Appl.*, vol. 57, pp. 270–284, Sept. 2016.
[48] B. Qu and P. N. Suganthan, "Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods," *Eng. Optimiz.*, vol. 43, no. 4, pp. 403–416, Apr. 2011.
[49] S. Zapotecas-Martinez and C. A. Coello Coello, "MONSS: A multi-objective nonlinear simplex search approach," *Eng. Optimiz.*, vol. 48, no. 1, pp. 16–38, 2016.
[50] T. T. Nguyen and X. Yao, "Continuous dynamic constrained optimization—the challenges," *IEEE Trans. Evol. Comput*., vol. 16, no. 6, pp. 769–786, Deb. 2012.
[51] H. K. Singh, A. Isaacs, T. T. Nguyen, T. Ray, and X. Yao, "Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 3127–3134.
[52] H. Richter. "Memory design for constrained dynamic optimization problems," *Applications of Evolutionary Computation*, pages 552–561, 2010.
[53] S. Yang and R. Tinos, "A hybrid immigrants scheme for genetic algorithms in dynamic environments," *Int. J. Autom. Comput.*, vol. 4, no. 3, pp. 243–254, Jul. 2007.
[54] S. Yang, "Genetic algorithms with memory-and elitism-based immigrants in dynamic environments," *Evol. Comput.*, vol. 16, no. 3, pp. 385–416, Sept. 2008.
[55] M. Mavrovouniotis and S. Yang, "Genetic algorithms with adaptive immigrants for dynamic environments," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 2130–2137.
[56] G. Ruan, G. Yu, J. Zheng, J. Zou, and S. Yang, "The effect of diversity maintenance on prediction in dynamic multi-objective optimization," *Appl. Soft Comput*., vol. 58, pp. 631–647, Sept. 2017.
[57] X. Ma, J. Yang, H. Sun, Z. Hu, and L. Wei, "Multiregional coevolutionary algorithm for dynamic multiobjective optimization," *Inf. Sci.*, vol. 545, pp. 1–24, Feb. 2021.
[58] K. Zhang, C. Shen, X. Liu, and G. G. Yen, "Multiobjective evolution strategy for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput*., vol. 24, no. 5, pp. 974–988, Oct. 2020.
[59] C. K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 103–127, Feb. 2009.
[60] D. Gong, B. Xu, Y. Zhang, Y. Guo, and S. Yang, "A similarity-based cooperative co-evolutionary algorithm for dynamic interval multiobjective optimization problems," *IEEE Trans. Evol. Comput*., vol. 24, no. 1, pp. 142–156, Feb. 2020.
[61] Z. Liang, T. Wu, X. Ma, Z. Zhu, and S. Yang, "A dynamic multiobjective evolutionary algorithm based on decision variable classification," *IEEE Trans. Cybern.*, vol. 52, no. 3, pp. 1602–1615, Mar. 2022.
[62] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 1999, pp. 1875–1882.
[63] W. Zhang, W. Zhang, G. G. Yen, and H. Jing, "A cluster-based clonal selection algorithm for optimization in dynamic environment," *Swarm Evol. Comput.*, vol. 50, pp. 100454, Nov. 2019.
[64] S. Y. Jiang and S. X. Yang, "A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 65–82, Feb. 2017.
[65] Y. G. Woldesenbet and G. G. Yen, "Dynamic evolutionary algorithm with variable relocation," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 500–513, Jun. 2009.
[66] R. Azzouz, S. Bechikh, and L. B. Said, "A dynamic multiobjective evolutionary algorithm using a change severity-based adaptive population management strategy," *Soft Comput.*, vol. 21, no. 4, pp. 885–906, Feb. 2017.
[67] Z. Liang, S. Zheng, Z. Zhu, and S. Yang, "Hybrid of memory and prediction strategies for dynamic multiobjective optimization," *Inf. Sci*., vol. 485, pp. 200–218, Jun. 2019.
[68] M. Jiang, Z. Huang, L. Qiu, W. Huang, and G. G. Yen, "Transfer learning-based dynamic multiobjective optimization algorithms," *IEEE Trans. Evol. Comput*., vol. 22, no. 4, pp. 501–514, Aug. 2018.
[69] M. Jiang, Z. Wang, H. Hong, and G. G. Yen, "Knee point based imbalanced transfer learning for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput*., vol. 25, no. 1, pp. 117–129, Feb. 2021.
[70] A. Gupta, Y. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Trans. Emerg. Topics Comput*., vol. 2, no. 1, pp. 51–64, Feb. 2018.
[71] M. Jiang, Z. Wang, L. Qiu, S. Guo, X. Gao, and K. C. Tan, "A fast dynamic evolutionary multiobjective algorithm via manifold transfer

learning," *IEEE Trans. Cybern*., vol. 51, no. 7, pp. 3417–3428, Jul. 2021.
[72] J. Li, T. Sun, Q. Lin, M. Jiang, and K. C. Tan, "Reducing negative transfer learning via clustering for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput*., vol. 26, no. 5, pp. 1102–1116, Oct. 2022.
[73] F Zou, G. G. Yen, and C Zhao, "Dynamic multiobjective optimization driven by inverse reinforcement learning," *Inf. Sci.*, vol. 575, pp. 468–484, Oct. 2021.
[74] F. Zou, G. G. Yen, L. Tang, and C. Wang, "A reinforcement learning approach for dynamic multi-objective optimization," *Inf. Sci.*, vol. 546, pp. 815–834, Feb. 2021.
[75] M. Rong, D. Gong, Y. Zhang, Y. Jin, and W. Pedrycz, "Multidirectional prediction approach for dynamic multiobjective optimization problems," *IEEE Trans. Cybern*., vol. 49, no. 9, pp. 3362–3374, Sept. 2019.
[76] L. Cao, L. Xu, E. D. Goodman, C. Bao, and S. Zhu, "Evolutionary dynamic multiobjective optimization assisted by a support vector regression predictor," *IEEE Trans. Evol. Comput*., vol. 24, no. 2, pp. 305–319, Apr. 2020.
[77] D. Xu, M. Jiang, W. Hu, S. Li, R. Pan, G. G. Yen, "An online prediction approach based on incremental support vector machine for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput*., vol. 26, no. 4, pp. 690–703, Aug. 2022.
[78] R. Rambabu, P. Vadakkepat, K. C. Tan, and M. Jiang, "A mixture-of-experts prediction framework for evolutionary dynamic multiobjective optimization," *IEEE Trans. Cybern*., vol. 50, no. 12, pp. 5099–5112, Dec. 2020.
[79] M. Rong, D. Gong, W. Pedrycz, and L. Wang, "A multimodel prediction method for dynamic multiobjective evolutionary optimization," *IEEE Trans. Evol. Comput*., vol. 24, no. 2, pp. 290–304, Apr. 2020.
[80] K. Zhang, C. Shen, X. Liu, and G. G. Yen, "Multiobjective evolution strategy for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput*., vol. 24, no. 5, pp. 974–988, Oct. 2020.
[81] F Zou, G. G. Yen, and L. Zhao, "A knee-guided prediction approach for dynamic multi-objective optimization," *Inf. Sci.*, vol. 509, pp. 193–209, Jan. 2020.
[82] T. T. Nguyen, X. Yao, "Benchmarking and solving dynamic constrained problems," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 690–697.
[83] Y. Wang, J. Yu, S. Yang, S. Jiang, S. Zhao, "Evolutionary dynamic constrained optimization: Test suite construction and algorithm comparisons," *Swarm Evol. Comput.*, vol. 50, pp. 100559, Nov. 2019.
[84] C. Bu, W. Luo, T. Zhu, R. Yi, and B. Yang. "Species and memory enhanced differential evolution for optimal power flow under double-sided uncertainties, " *IEEE Trans. Sustain. Comput*., vol. 5, no. 3, pp. 403–415, Jul. 2020.
[85] C. Bu, W. Luo, and L. Yue. "Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies," *IEEE Trans. Evol. Comput*., vol. 21, no. 1, pp. 14–33, Feb. 2017.
[86] A. Konak, D. W. Coit, A. E. Smith, "Multi-objective optimization using genetic algorithms; A tutorial," *Reliab. Eng. Syst. Saf*., vol. 91, no. 9, pp. 992-1007, Sep. 2006.
[87] Q. Chen, J. Ding, S. Yang, and T. Chai, "Constrained Operational Optimization of a Distillation Unit in Refineries with Varying Feedstock Properties," *IEEE Trans. Control Syst. Technol*., vol. 28, no. 6, pp. 2752–2761, Nov. 2020.
[88] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst*., vol. 9, pp. 115-148, Apr. 1995.
[89] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[90] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multiobjective evolutionary algorithm," in *Proc. Mexican Int. Conf. Artif. Intell.*, 2004, pp. 688–697.
[91] S. Jiang and S. Yang, "Evolutionary dynamic multiobjective optimization benchmarks and algorithm comparisons," *IEEE Trans. Cyber.*, vol. 47, no. 1, pp. 198–211, Jan. 2017.
[92] D. Yazdani, J. Branke, M. N. Omidvar, T. T. Nguyen, and X. Yao, "Changing or keeping solutions in dynamic optimization problems with switching costs," in *Proc. Annu. Conf. Genet. Evol. Comput*. (*GECCO*), Madrid, Spain, 2018, pp. 1095–1102.
[93] D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 143–155, Feb. 2019.
[94] A. Muruganantham, K. C. Tan, and P. Vadakkepat. "Evolutionary dynamic multiobjective optimization via Kalman filter prediction." *IEEE Trans. Cyber*., vol. 46, no. 12, pp. 2862–2873, Dec. 2016.

Qingda Chen (S'19–M'20) received the B.S. degree from Yantai University, Yantai, China, in 2013, and the Ph.D. degree in control theory and control engineering from the State Key Laboratory of Synthetical Automation for Process Industry, Northeastern University, Shenyang, China, in Apr. 2020.

He is currently a Lecturer with the State Key Laboratory of Synthetical Automation for Process Industry, Northeastern University, Shenyang, China. His current research interests include computational intelligence and its application in the industrial processes.

Jinliang Ding (M'09–SM'14) received the Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2012.

He is currently a Professor with the State Key Laboratory of Synthetical Automation for Process Industry, Northeastern University. He has authored or coauthored more than 100 refereed journal and conference articles. He is also the inventor or co-inventor of over 20 patents. His current research interests include modeling, plant-wide control and optimization for the complex industrial systems, machine learning, industrial artificial intelligence, and computational Intelligence and application.

Dr. Ding was the recipient of the Young Scholars Science and Technology Award of China in 2016 and the National Science Fund for Distinguished Young Scholars in 2015, and the National Technological Invention Award in 2013. One of his paper published on *Control Engineering Practice* was selected for the Best Paper Award of 2011–2013.

Gary G. Yen (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1992.

He was with the Structure Control Division, U.S. Air Force Research Laboratory, Albuquerque, NM, USA. In 1997, he joined Oklahoma State University (OSU), Stillwater, OK, USA, where he is currently a Regents Professor with the School of Electrical and Computer Engineering.

His research interests include intelligent control, computational intelligence, conditional health monitoring, signal processing, and their industrial/defense applications. From 2006 to 2009, he was the Founding Editor-in-Chief of the IEEE Computational Intelligence Magazine. He is currently an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE, IEEE TRANSACTIONS ON EMERGING TOPICS ON COMPUTATIONAL INTELLIGENCE, and IEEE TRANSACTIONS ON CYBERNETICS.

Shengxiang Yang (M' 00–SM' 14) received the B.Sc. and M.Sc. degrees in automatic control and the Ph.D. degree in systems engineering from Northeastern University, Shenyang, China in 1993, 1996, and 1999, respectively.

His current research interests include evolutionary and genetic algorithms, swarm intelligence, computational intelligence in dynamic and uncertain environments, artificial neural networks for scheduling, and relevant real-world applications.

Prof. Yang serves as an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON CYBERNETICS etc.

Tianyou Chai (M'90–SM'97–F'08) received the Ph.D. degree in control theory and engineering from Northeastern University, Shenyang, China, in 1985.

He became a Professor with Northeastern University in 1988 and a Chair Professor in 2004. His current research interests include adaptive control, intelligent decoupling control, integrated plant control and systems, and the development of control technologies with applications to various industrial processes.

Prof. Chai is a member of the Chinese Academy of Engineering, an Academician of the International Eurasian Academy of Sciences, and an IFAC Fellow.