A Novel Evolutionary Algorithm for Dynamic Constrained Multiobjective Optimization Problems

Qingda Chen, Student Member, IEEE, Jinliang Ding, Senior Member, IEEE,
Shengxiang Yang, Senior Member, IEEE, and Tianyou Chai, Fellow, IEEE

Abstract—To promote research on dynamic constrained multiobjective optimization, we first propose a group of generic test problems with challenging characteristics, including different modes of the true Pareto front (e.g., convexity–concavity and connectedness–disconnectedness) and the changing feasible region. Subsequently, motivated by the challenges presented by dynamism and constraints, we design a dynamic constrained multiobjective optimization algorithm with a nondominated solution selection operator, a mating selection strategy, a population selection operator, a change detection method, and a change response strategy. The proposed test problems can be used to clearly distinguish the performance of algorithms, and that the proposed algorithm is very competitive for solving dynamic constrained multiobjective optimization problems in comparison with state-of-the-art algorithms.

Index Terms—Dynamic constrained multiobjective optimization, test problems, population selection, change response

I. INTRODUCTION

ANY real-world optimization problems, particularly online optimization problems or optimal control problems, involve multiple objectives and constraints that may change over time throughout the optimization [1]; such problems can be called dynamic constrained multiobjective optimization problems (DCMOPs). Without loss of generality, a DCMOP can be mathematically formulated as follows:

\[
\begin{align*}
\min & \quad F(x_t) = (f_1(x_t), \ldots, f_m(x_t)) \\
\text{s.t.} & \quad h_k(x_t) = 0, \quad k = 1, \ldots, h \\
& \quad g_k(x_t) \geq 0, \quad k = 1, \ldots, g \\
& \quad x = [x_1, \ldots, x_j, \ldots, x_n] \in \Omega \\
& \quad x_j^{\min} \leq x_j \leq x_j^{\max}
\end{align*}
\]

where \( m \) is the objective function index, and \( M \) is the number of objective functions, \( j \) is the decision variable index, and \( n \) is the number of decision variables. \( x \) is a solution that consists of \( n \) decision variables, and \( x_j \) is the \( j \)th decision variable, which is bounded by corresponding lower (\( x_j^{\min} \)) and upper (\( x_j^{\max} \)) bounds. \( \Omega \) is the decision space, and \( t \) represents a dynamic environment. \( h \) and \( g \) are the numbers of dynamic equality and inequality constraints, respectively. \( F(x_t) \) is the objective function vector to be minimized at \( t \), and \( f_m(x_t) \) denotes the \( m \)th objective function. \( h(x_t) \) and \( g(x_t) \) represent the \( k \)th equality and inequality constraints, respectively, that vary with \( t \).

If a solution satisfies all constraints, then it is called a feasible solution; otherwise, it is called an infeasible solution. For two feasible solutions \( x_1 \) and \( x_2 \) of a DCMOP at \( t \), \( x_1 \) is said to dominate \( x_2 \) if and only if \( f_m(x_1, t) \leq f_m(x_2, t) \) for every \( m \in \{1, \ldots, M\} \) and \( f_d(x_1, t) \leq f_d(x_2, t) \) for at least one index \( b \in \{1, \ldots, M\} \). If there is no other solution in \( \Omega \) to dominate \( x_1 \), then \( x_1 \) is a nondominated solution (i.e., a Pareto-optimal solution). \( F(x_1, t) \) is then called a Pareto-optimal objective vector at \( t \). The set of all Pareto-optimal solutions is called the Pareto-optimal set (POS). Similarly, the set of all Pareto-optimal objective vectors is called the Pareto-optimal front (POF).

The goal when solving a multiobjective optimization problem (MOP) is to obtain a set of nondominated solutions, all of which must be feasible. Compared with constrained MOPs without dynamism (CMOPs) and dynamic MOPs without constraints (DMOPs), DCMOPs are more challenging due to the simultaneous presence of constraints and dynamism. Specifically, a change is usually not observable and may even be unknown, and it may arise in either the objective functions or constraints. Detecting environmental changes is difficult if they are unforeseen in DCMOPs. The feasible region of objective functions (hereafter referred to simply as the feasible region) is affected by constraints and may increase or decrease with dynamism, changing the number of feasible solutions. Abandoning infeasible solutions may be beneficial for population convergence. However, some previous infeasible solutions close to the boundary of the feasible region can easily become feasible or even nondominated in new environments.

This paper has supplementary material.
Therefore, determining how to best handle infeasible solutions is a considerable challenge. For a DCMOP, the true POF of the objective functions (hereafter called the true POF) may be determined by the unconstrained POF (i.e., the POF of a DCMOP without constraints) and bounds of the feasible region. The true POF of a DCMOP may become disconnected when the lower bound of the feasible region is irregular. An example of the true POF of a DCMOP with two objectives (i.e., \( f_1(x, t) \) and \( f_2(x, t) \)) is given in Fig. 1, and its functions are given in Section S-I of the supplementary material. The disconnectedness of the true POF may cause the population to get stuck at local regions, decreasing the population diversity. Thus determining how to design the population update strategy that can quickly track the true POF and improve population diversity is a challenging task. Furthermore, dynamism and constraints can cause the true POF to move in an irregular manner. Hence, designing an effective change response strategy that can track the new POF is an arduous task.

In recent years, many contributions have been made regarding attempts to solve CMOPs and DMOPs in several important aspects, including benchmark problems [2]–[7], [63], performance metrics [2], [8]–[11], and algorithms [12]–[23]. It is worth noting that the moving peak benchmark problem designed in [63] transforms the single objective problem proposed in [56] into a two-objective problem by assigning a random objective value into each individual, which is a good attempt. However, many real-world problems are DCMOPs, and the current states of both benchmark problem and algorithm designs for DCMOPs are considerably less advanced compared to those for CMOPs and DMOPs [24]. Azzouz et al. [25] designed a set of test instances for DCMOPs and proposed a dynamic constrained algorithm to solve them. However, the true POSs in the designed test problems show only small variations between \( t \) and \( t+1 \), making it difficult to test the change response capability of an algorithm. Azzouz et al. [24] proposed a dynamic constrained multiobjective evolutionary algorithm (DC-MOEA) for solving the test problems designed in [25]. However, this algorithm does not update the retained previous feasible solutions when the environment changes, resulting in a slow tracking ability when the change in the true POF is large.

To advance the state of research on DCMOPs, we design a set of test problems and propose a novel dynamic constrained multiobjective evolutionary algorithm (dCMOEA) to solve them. The main contributions of this work are as follows:

1) Following a few basic design principles proposed in [12] and [56], two important characteristics of DCMOPs are introduced into the test problems. Specifically, the true POF in each proposed test problem is simultaneously determined by the unconstrained POF and bounds of the feasible region, causing the true POF to switch between different modes (e.g., connected and disconnected). This characteristic is useful for testing an algorithm’s convergence speed, reactivity, and rapid tracking capability with respect to the new true POF. The feasible region in each of the proposed test problems can increase or decrease with dynamism in an irregular manner, meaning that the number of infeasible solutions and even the true POF of a DCMOP can change. The second characteristic makes it possible to test an algorithm’s capability of handling infeasible solutions.

2) The nondominated solutions in DCMOPs usually vary with dynamism and constraints, and some previous nondominated solutions may become infeasible or dominated in a new environment. Based on this fact, we propose a nondominated solution selection operator that is suitable for DCMOPs. When a new environment arises, the proposed nondominated solution selection operator can reserve a part of nondominated solutions obtained in the previous environment, increasing the diversity of the nondominated set in a new environment.

3) For handling the infeasible solutions and improving population diversity, the constraint handling technique proposed in [15] is introduced into the designed mating selection strategy and population selection operator. Considering that the feasible region may vary with dynamism, the population selection operator integrates the information of the obtained feasible solutions. The proposed operator can adaptively update the population, balancing the trade-off between the population diversity and convergence capability of dCMOEA.

4) For a DCMOP in which the environmental changes are unforeseen, we propose a change detection strategy that can accurately detect dynamism arising in either objective functions or constraints. dCMOEA reinitializes a portion of the solutions and reuses some previous solutions with competitive performance (i.e., small objective function and constraint violation values) when the environment changes. Meanwhile, we exploit information collected from the reinitialized solutions and an approximate feasibility ratio to update the retained previous solutions, allowing the proposed algorithm to quickly track the new POF.

The remainder of this paper is organized as follows. Section II discusses some related work on the constraint handling techniques used for CMOPs and the change response strategies used for DMOPs. Section III presents the design of the test problems. Section IV describes the framework of the proposed dCMOEA, together with detailed descriptions of each component of the algorithm. Performance metrics and a comprehensive comparison of various multiobjective evolutionary algorithms (MOEAs) on the test problems are presented in Section V. Section VI offers a further discussion of the proposed algorithm. Section VII outlines some conclusions and suggests directions for future research.

II. RELATED WORK

Considering that there are few works on solving DCMOPs, in this section, we first present some recent studies on constraint handling techniques for CMOPs and then discuss the
change response strategies adopted in DMOPs.

A. Constraint Handling Techniques

The constraint handling techniques for CMOPs can be divided into three categories [21]: techniques that place a higher priority on feasible solutions to survive to the next generation, techniques for balancing the trade-off between feasibility and convergence, and techniques for repairing infeasible solutions.

The first category is based on a preference for feasible solutions, which are considered better than infeasible ones. The constraint handling method proposed by Coello Coello and Christiansen [26] simply ignores infeasible solutions, which results in difficulty when solving problems with a narrow feasible region. Deb et al. [13] designed a constrained dominance relation in which a solution \( x_1 \) is said to dominate \( x_2 \) if 1) the former is feasible while the latter is not, 2) they are both infeasible, but \( x_1 \) has a smaller overall constraint violation, or 3) both of them are feasible and \( x_1 \) dominates \( x_2 \). Following a similar idea, Cheng et al. [27] updated the population on the basis of the degree of constraint violation. Fonseca and Fleming [28] designed a unified framework for solving CMOPs in which a higher priority is assigned to the constraints than to the objective functions, thus the search process prioritizes finding feasible solutions over finding optimal ones.

Takahama et al. [29] and Asafuddoula et al. [30] proposed an \( e \)-constrained dominance relation in which the constraint violation values of two solutions are not considered if they are smaller than a certain threshold. In the method of [31], infeasible solutions do not survive to the next generation if the number of feasible solutions is sufficient. Fan et al. [32] designed an angle-based constrained dominance principle in which \( x_1 \) dominates \( x_2 \) if the former is feasible while the latter is not. For a DCMOP, the feasible region can increase or decrease with dynamism. Giving feasible solutions a higher priority is beneficial for speeding up the convergence of an algorithm when the feasible region is large. However, emphasizing the importance of feasibility may reduce the population diversity.

Techniques in the second category try to balance the trade-off between feasibility and convergence during the evolutionary process. Angantyr et al. [33] and Young [34] developed a constrained dominance relation by combining the ranks of a solution based on its objective functions and constraint violation values. Woldesenbet et al. [15] proposed a new constraint handling technique in which each objective function of a solution is modified in accordance with its original objective function values and constraint violation values. The dominance relation is determined on the basis of the modified objective functions and the nondominated sorting procedure proposed in the nondominated sorting genetic algorithm II (NSGA-II [13]). To improve the population diversity, Li et al. [35] developed a method in which the worst solution is given a second chance for survival when it is associated with an isolated subregion. To utilize useful information included in infeasible solutions, Peng et al. [36] introduced infeasible weights, which change with smaller constraint violation values and better objective function values, to maintain many well-diversified infeasible individuals. Ning et al. [37] proposed a constrained nondominated sorting rank approach in which each solution is associated with a constrained nondomination rank in accordance with its Pareto rank and constraint rank. Sorkhabi et al. [38] designed an efficient approach for constraint handling in which infeasible particles are evolved in the constraint region toward feasibility, and feasible particles are evolved toward Pareto optimality. However, for a DCMOP, a smaller feasible region may lead to a reduction in the number of feasible solutions. In this case, removing some feasible solutions may cause the algorithm to fail to converge because the solutions in the POS are feasible.

Techniques in the last category aim to repair infeasible solutions. In [39], Harada et al. proposed a constraint handling method called Pareto descent repair, which incorporates the gradient projection method. However, the gradient information for DCMOPs is usually unavailable. Sigh et al. [40] used an approximate descent direction method to reduce the degree of constraint violation. Jiao et al. designed a feasible-guiding strategy that aims to find feasible solutions close to the feasible region in a feasible direction with the help of infeasible solutions [41]. For a DCMOP, however, the feasible region varies irregularly with dynamism, and it is difficult to find a good feasible direction in which to repair infeasible solutions.

Additionally, there are some effective constraint handling strategies to solve the constraints in single objective optimization problems. Runarsson and Yao [58] proposed a stochastic ranking in which a probability is introduced to balance the objective and overall constraints violation stochastically. This approach can significantly improve the optimization performance without any special constrain handling operator. In [59], they designed an improved stochastic ranking in which comparison between two solutions may be based on the overall constraint violation alone or objective value alone as randomly determined. Thus, some infeasible solutions with better value may be selected in evolution. Allmendinger and Knowles [60] investigated two interesting types of ephemeral resource constraints (ERCs): one encodes periodic resource availabilities, and the other models ‘commitment’ constraints that make the evaluable part of the space a function of earlier evaluations conducted. The studies on both types of constraints are of great significance to real-world applications, especially closed-loop optimization settings. Note that ERCs are not standard constraints but restrictions on a series of solutions that can be actually evaluable at a given time during the optimization, arising because of resourcing issues [61].

B. Change Response Strategies

The strategies for responding to changes can be divided into the following three categories.

1) Diversity Enhancement: These methods increase the population diversity by means of certain methodologies when a change is detected. Woldesenbet and Yen [42] proposed a dynamic EA with variable relocation that relocates solutions based on the changes in the objective function values due to a change in the environment and the average sensitivities of their decision variables to the corresponding changes in the objective space. Yang and Tinos [43] and Mavrovouniotis and Yang [45] proposed a hybrid immigrant scheme based on memory-based immigrants [44] and elitism-based immigrants, and these methods are effective in dealing with changing DMOPs.

2) Memory Mechanism: These mechanisms reuse the past
information to improve the performance of EAs [56]. Goh and Tan [11] proposed an idea based on comparing the potential of new regions with past information to decide whether a subpopulation should be initialized when an environmental change occurs. Specifically, the particular subpopulation must be reinitialized in the space from which the winner is sampled. Wang and Li [46] designed a reinitialization strategy to respond to environmental changes. In their reinitialization strategy, the initialized solutions are chosen from the current population and archived solutions with a probability of 0.2. In [47], Azzouz et al. proposed generating some solutions via a memory-based strategy that makes use of previous optimal solutions.

3) Prediction Strategy: Such a strategy usually predicts the state of the new environment using existing information and other learning techniques. Considering the properties of continuous DMOPs, Zhou et al. proposed a population prediction strategy that is divided into two parts: a center point and a manifold [16]. A sequence of center points is used to predict the next center, and the previous manifolds are maintained to estimate the next manifold.

The three types of change response strategies discussed above perform well for solving different DMOPs. Nevertheless, for DCMOPs, the feasible region move with dynamism, and a previously feasible region may become infeasible in a new environment. In other words, previously feasible solutions may become infeasible. Therefore, a change response strategy must be designed with proper consideration of the constraints.

III. PROPOSED TEST PROBLEMS

Test problems play a crucial role in judging whether an algorithm is a candidate for solving MOPs [3]–[4]. Currently, test instances for DMOPs [3]–[5], [63] and CMOPs [6], [31], [48] have been proposed. However, these test problems do not consider dynamism and constraints simultaneously. Analogous to the design principle of dynamic single-objective test problems proposed in [56], the test problems of a DCMOP should also be close to reality, easy to describe, simple, and easy to analyze. In real-world applications of DCMOPs, the objective functions and the feasible region may both change with dynamism, changing the true POF. Specifically, the bounds of the feasible region may be determined by the unconstrained POF and constraints, and its true POF is a part of the lower boundary of the feasible region. Dynamism and constraints can change the lower boundary, causing the true POF to alternate between connectedness and disconnectedness as well as between convexity and concavity, which pose a tremendous challenge for the convergence speed and response capability of an algorithm. The feasible region may move with dynamism, resulting in changes to the feasible solutions, infeasible solutions, and nondominated solutions. Some previously nondominated solutions may become dominated or even infeasible in the new environment. Meanwhile, some previously infeasible solutions may become feasible or even nondominated. This possibility poses a challenge regarding an algorithm’s capability of handling infeasible solutions. Based on these characteristics arising in real-world applications, this paper considers the four types of test problems, and the real-world problems related to these four types are discussed in Section S-II of the supplementary material.

Type I: The feasible region first increases with $t$ and then decreases, while the true POF changes from continuous to disconnected and finally back to continuous.

Type II: The feasible region first decreases with $t$ and then increases, while the true POF changes from disconnected to continuous and finally back to disconnected.

Type III: The feasible region first increases with $t$ and then decreases, while the true POF changes from disconnected to continuous and finally back to disconnected.

Type IV: The feasible region first decreases with $t$ and then increases, while the true POF changes from continuous to disconnected and finally back to continuous.

Considering that the objective functions in the electric power supply problem studied in [49] oscillate among several optimization modes, this study considers two cases for the unconstrained POF of a DCMOP, called Case 1 and Case 2.

Case 1: The unconstrained POF has one mode that includes convexity and concavity. Algorithms with fast convergence can easily solve such problems.

Case 2: The objective functions oscillate among several modes that may include convexity and concavity.

For the real-world optimization problems especially for the optimization of electric energy (e.g., the dynamic power supply problem in magnesia grain manufacturing [49], the peaking shaving and valley filling problem in plug-in electric vehicles [62]), their objective functions and constraints may include trigonometric functions. Additionally, constraints in real-world problems usually cause the irregularity of feasible region, leading to the irregularity of the true POF in DCMOPs. Designing test problems with trigonometric functions can easily reflect these characteristics arising in real-world problems by adjusting the corresponding parameters. Based on the above discussions and benchmarks designed in [4], [6], and [48], this study proposes the following instance generator.

\[
\begin{align*}
\text{min} & \quad F(x, t) = (f_1(x, t), f_2(x, t))^T \\
\text{obj} & \quad f_1(x, t) = (1 + g(x_{S1}, t))h(x_{II}, t) + A_1 \sin(W_1 x_{h}(x_{II}, t))) \\
& \quad f_2(x, t) = (1 + g(x_{S1}, t))h(x_{I}, t) - h(x_{II}, t) + A_2 \sin(W_2 \pi h(x_{II}, t))) \\
& \quad \cos(\theta) \times f_2(x_{I}, t) - e - \sin(\theta) \times f_1(x_{II}, t) - m(t) \geq 0 \\
& \quad s.t. \quad e \times \sinh(\pi x_{I}) \times f_2(x_{I}, t) - e \times \cos(\theta) \times f_1(x_{II}, t) \geq 0 \\
& \quad s \times f_2(x_{I}, t) + f_2(x_{II}, t) - s(t) < 0 \\
& \quad z(t) > m(t)
\end{align*}
\]

In the objective functions, $s(t)$ is a nonnegative dynamic parameter that controls the movement of the unconstrained POF. A large movement distance of $s(t)$ causes the POF to also exhibit a large movement. $x_I$ and $x_{II}$ are subvectors of $x$. $h(x_{II}, t)$ is a nonnegative function such that $0 \leq h(x_{II}, t) \leq s(t)$. $A_1$ controls the curvature of the unconstrained POF, and $W_1$ determines the number of mixed concave and convex segments of the unconstrained POF. Setting $W_1$ properly can make the unconstrained POF oscillate among several modes. $g(x_{S1}, t)$ is a nonnegative function that determines the starting position and change of solutions in POS. The minimum value of $g(x_{S1}, t)$ is zero, so the objective functions can be formulated as follows:

\[
\begin{align*}
f_1 + f_2 = s(t) + 2A_2 \sin(W_1 \pi x_{II}, t) \frac{f_2 - f_1 + s(t)}{2}
\end{align*}
\]

Based on Equation (3), the unconstrained POFs with different settings for $s(t), A_1,$ and $W_1$ are shown in Fig. 2.

In the constraints of Equation (2), the effects of parameters $\theta,$
Fig. 2. The unconstrained POFs of test problems with different overall shapes.  

where $m(t)$ and $z(t)$ are nonnegative dynamic parameters that control the lower and upper bounds, respectively, of the feasible region determined by constraints. 

To embody the characteristic changes for problems of Types I-IV, $s(t)$, $m(t)$, and $z(t)$ are set as follows:

$$
\begin{align*}
    s(t) &= \alpha - \beta \times t \\
    m(t) &= \nu - \chi \times t \\
    z(t) &= \delta - \lambda \times t
\end{align*}
$$

(4)

where $\alpha$ controls the starting position of the unconstrained POF; $\nu$ and $\delta$ control the starting positions of the lower and upper bounds, respectively, of the feasible region; $\beta$ controls the step length of the unconstrained POF; and $\chi$ and $\lambda$ control the step lengths of the lower and upper bounds, respectively, of the feasible region.

For Cases 1 and 2, one can select between these two cases by setting $W_t$ as desired.

Case 1: The unconstrained POF has one mode that includes convexity and concavity, with $W_t$ set to a constant.

Case 2: The unconstrained POF oscillates among several optimization modes that may include convexity and concavity, with $W_t = \sin(\eta \pi (t+1))$. Note that $\varphi$ controls the curvature of oscillation, while $\eta$ determines the period of oscillation (i.e., the period is $2\pi / \eta \pi$). Two diagrams are given in Fig. 3 to illustrate the impacts of $\varphi$ and $\eta$ on the unconstrained POFs. Their objective functions are given in Equations (5) and (6), respectively.

$$
\begin{align*}
    f_1 + f_2 &= 3.64 - 0.07t + 0.1(\sin(\varphi \pi \pi (t+1)))(\frac{f_0 - f_1 + 3.64 - 0.07t}{2}) \\
    f_1 + f_2 &= 3.64 - 0.07t + 0.1(\sin(\varphi \pi \pi (t+1)))(\frac{f_0 - f_1 + 3.64 - 0.07t}{2})
\end{align*}
$$

(5)

The ideas on scaling up the proposed test problems are discussed in Section S-III of the supplementary material.

IV. PROPOSED DCMOEA

In a DCMOP, the true POF and feasible region vary with dynamism. To handle infeasible solutions and respond to changes, this paper designs an algorithm called dCMOEA, whose basic framework is presented in Algorithm 1.

As shown in Algorithm 1, dCMOEA starts with an initial population $P$. Then, the nondominated solution set $A$ is determined (line 4 of Algorithm 1). In the each generation, if a new environment is detected, then a change response strategy (line 8 of Algorithm 1) is used to update $P$, and a nondominated solution selection operator (line 9 of Algorithm 1) is applied to update $A$. Afterwards, a mating selection operator (line 11 of Algorithm 1) is used to select two parents (i.e., $p_{i1}$ and $p_{i2}$) for generating the $i$th child $q_i$ (line 12 of Algorithm 1). Subsequently, $q_i$ is stored in an offspring set $Q$. At the end of each generation, a population selection operator (line 14 of Algorithm 1) is applied to update $P$. Finally, a nondominated solution selection operator (line 15 of Algorithm 1) is used to update $A$. The above procedure is executed until the termination condition is met. In the following sections, each component of dCMOEA will be described in detail.

A. Initialization

In dCMOEA, the evolutionary process starts with an initial population $P$ with $N$ solutions. For a DCMOP, it is difficult to generate a feasible solution when the feasible region is small. However, the solutions in the POS and POF must be feasible, so this paper stipulates that there must be at least one feasible solution in the initial population to prevent the absence of nondominated solutions. During initialization, we first use Equation (7) to generate the components of each solution.

$$
\begin{align*}
    x_{i,j} &= x_{i,j}^{\text{min}} + rand \times (x_{i,j}^{\text{max}} - x_{i,j}^{\text{min}})
\end{align*}
$$

(7)

where $rand$ is a random number in the range $[0, 1]$.

Subsequently, the constraint deviation values of each solution are calculated using Equations (8) and (9).

$$
\begin{align*}
    G_k(x, t) &= \max \{0, -g_k(x, t)\}, \quad 1 \leq k \leq n_g \\
    H_k(x, t) &= \max \{0, h_k(x, t) - c_k\}, \quad 1 \leq k \leq n_h
\end{align*}
$$

(8)
where $G_i(x_i, t)$ and $H_i(x_i, t)$ denote the $i$th inequality and equality constraint violations, respectively, at $t$. The parameter $c$ is a tolerance value for relaxing the equality constraints to inequality constraints, and $\nu(x_i, t)$ is the total constraint violation of $x_i$ at $t$. $x_i$ is a feasible solution if $\nu(x_i, t)$ is zero.

If there is no feasible solutions, then the initial population is regenerated until there is at least one feasible solution.

B. Nondominated Solution Selection

The nondominated solution selection procedure aims to preserve nondominated solutions from the Population $P$ and $A$. For a DCMOP, some solutions in $A$ may become dominated solutions or even infeasible solutions when a new environment arises. To obtain nondominated solutions with diversity, we propose that, when a change is detected, the infeasible and dominated solutions in $A$ are first removed, followed by the infeasible and dominated solutions in $P$. Afterward, $A$ is updated according to the dominance relationship between solutions in $A$ and the nondominated solutions in $P$. Note that if the environment changes, then the infeasible and dominated solutions in $P$ are removed, and $A$ is updated according to the dominance relationship between solutions in $A$ and the nondominated solutions in $P$. Specifically, when a new environment is detected, the constraint deviation values of all solutions in $A$ are calculated using Equations (8) and (9), and solutions with constraint deviation values larger than zero are removed from $A$. Afterward, each solution in $A$ is associated with a fitness value, which can be calculated via Equation (10).

$$F(i) = \left| \left\{ i' \in R : i' \prec i \right\} \right|$$

where $R$ is the set of solutions, and $i$ and $i'$ are the indices of two different solutions in $R$. $\left| \cdot \right|$ represents the cardinality of a set, and $i' \prec i$ indicates that $i'$ dominates $i$. $F(i)$ is the fitness value of the $i$th solution in $R$.

Subsequently, the dominated solutions in $A$ are removed. After it, there are two cases for $A$: one is that $A$ becomes an empty set, and another is that there is at least one solution in $A$.

To preserve nondominated solutions from the population $P$, the constraint deviation values of each solution in $P$ are calculated using Equations (8) and (9). Afterward, all feasible solutions in $P$ are copied into an empty set $S$. Each solution in $S$ is associated with a fitness value according to Equation (10). All nondominated solutions in $S$ are copied into an empty set $S'$. If $A$ is empty, then all solutions in $S'$ are copied into $A$. If there is at least one solution in $A$, then the dominance relations between the solutions in $A$ and $S'$ are assessed, and the following three cases are considered:

1) If a solution in $S'$ is dominated by any solution in $A$, then this solution in $S'$ is not considered.
2) If a solution in $S'$ is not dominated by any solution in $A$ and does not dominate any solution in $A$, then this solution in $S'$ is added to $A$.
3) If a solution in $S'$ is not dominated by any solution in $A$ and dominates some solutions in $A$, then all solutions dominated by this solution in $S'$ are removed from $A$.

Last, some solutions in $A$ must be removed if $|A|$ is greater than $N$. In this paper, the crowding distance operator applied in NSGA-II is used to perform a truncation operation to ensure that $|A|$ is equal to $N$. The pseudocode for nondominated solution selection is presented in Algorithm 2.

```
Algorithm 2 Non-dominated solution selection
1: Input: $P, A; S = \Phi, S' = \Phi$
2: Output: $A$
3: if a new environment is detected then
4:     for $i = 1$ to $|P|$
5:         Calculate constraint deviation value $\nu(x_i, t)$ of the $i$th solution in $A$
6:     end if
7:     if $\nu(x_i, t)$ is not zero then
8:         Remove the $i$th solution from $A$
9:     end if
10: end for
11: Calculate the fitness value of the $i$th solution in $A$ using Equation (10)
12: if $A = \Phi$ then
13:     Remove solutions having a fitness value that is more than one from $A$
14: end if
15: for $i = 1$ to $|S|$
16:     Calculate constraint deviation value $\nu(x_i, t)$ of the $i$th solution in $P$
17:     if $\nu(x_i, t)$ is zero then
18:         Add the $i$th solution into $S'$
19:     end if
20: end for
21: Calculate fitness values of solutions in $S$ using Equation (10)
22: Copy all solutions having a fitness value of zero in $S$ to an empty set $S'$
23: if $A = \Phi$ then
24:     All solutions in $S'$ are copied into $A$
25: else
26:     for $i = 1$ to $|S'|$
27:         if the $i$th solution in $S'$ is dominated by anyone in $A$ then
28:             $A$ remains unchanged;
29:         else
30:             if the $i$th solution in $S'$ is not dominated by any one in $A$ then
31:                 The $i$th solution in $S'$ is added into $A$
32:             else
33:                 All solutions in $A$ dominated by $i$th solution are removed;
34:         end if
35:     end if
36: end for
37: end if
```
modification method includes a distance measure and an adaptive penalty. The detailed modification procedure in calculating the distance measure is as follows:

First, dCMOEA normalizes the objective values by Equations (11)-(13).

\[ f_{m, i}^{\max} = \max_{x \in P} f_m(x, t) \]  

\[ f_{m, i}^{\min} = \min_{x \in P} f_m(x, t) \]  

\[ f_m(x, t) = f_m(x, t) - f_m^{\min} \]  

where \( f_m^{\max} \) and \( f_m^{\min} \) denote the maximum and minimum of the \( m \)th objective function value in \( F(x, t) \) at \( t \), respectively.

\( f_m(x, t) \) is the \( m \)th normalized objective function value of \( x \) at \( t \).

The total normalized constraint violation of a candidate solution at \( t \) is then calculated by Equations (8), (9), and (14):

\[ \nu(x, t) = \frac{1}{n_g} \sum_{k=1}^{n_g} G_k(x, t) + \frac{1}{n_h} \sum_{k=1}^{n_h} H_k(x, t) \]  

where \( \nu(x, t) \) is the total constraint violation of \( x \) at \( t \). It is worth noting that the smaller the value of \( \nu(x, t) \) is, the better the candidate solution.

The “distance” value of \( x \) in the \( m \)th objective function (denoted as \( d_m(x, t) \)) is obtained by Equation (15).

\[ d_m(x, t) = \sqrt{f_m(x, t)^2 + \nu(x, t)^2} \]  

The “penalty” value of \( x \) in the \( m \)th objective function (denoted as \( p_m(x, t) \)) is obtained by Equation (16).

\[ p_m(x, t) = (1 - r_J) \times \nu(x, t) + r_J \times Y_m(x, t) \]

where \( r_J \) denotes the feasible ratio of the current population, 

\[ r_J = \frac{\text{number of feasible solutions in current population}}{\text{population size}} \]

\[ Y_m(x, t) = \begin{cases} 0, & \text{if } \nu(x, t) = 0 \\ f_m(x, t), & \text{otherwise} \end{cases} \]

The \( m \)th modified objective value of \( x \) is formulated as follows:

\[ f_m'(x, t) = d_m(x, t) + p_m(x, t) \]

\[ f_m'(x, t) = \begin{cases} d_m(x, t) + p_m(x, t), & \text{if } f_m'(x, t) \leq p_m(x, t) \\ p_m(x, t), & \text{otherwise} \end{cases} \]

After the objective function values of \( x \) are modified, each solution is assigned a fitness value in accordance with Equation (10), and crowding distances are calculated using the method proposed in NSGA-II. Then choosing two parents (i.e., \( p_1 \) and \( p_2 \)) for the \( i \)th child employs the mating selection operator according to the fitness values and crowding distances of the solutions in \( P \). The pseudocode for mating selection is presented in Algorithm 3.

Once a parent population has been constructed, the popular simulated binary crossover (SBX) and polynomial mutation (PM) operators proposed in [54] are used to generate offspring.

D. Population Selection Operator

For a DCMP, if the feasible region decreases with dynamism, then more infeasible solutions may arise. Conversely, the number of infeasible solutions may decrease. Therefore, handling infeasible solutions while considering dynamism is key to obtaining a suitable set of approximations.

Simply discarding infeasible solutions might cause the algorithm to become trapped in local optima [15] whereas retaining more infeasible solutions may reduce the convergence speed of the algorithm. Note that dCMOEA must have a high convergence speed to allow it to track the POF quickly in the case of fast and frequent changes. To balance the convergence speed of the algorithm and the diversity of the population, dCMOEA considers two cases for handling infeasible solutions: the case where the number of feasible solutions of in \( P \) and \( Q \) is less than or equal to the threshold \( NF \), and the case where the number of feasible solutions is more than \( NF \).

a) The number of feasible solutions is fewer than or equal to \( NF \). For convenience of description, the solutions in \( P \) and \( Q \) are copied to an empty set \( C \). The feasible solutions in \( C \) are allocated to a feasible set \( FC \), and the infeasible ones are allocated to an infeasible set \( IC \). Considering that the solutions in the POS must be feasible, we propose that all solutions in \( FC \) should be retained for the next generation, and the remaining \( N-|FC| \) solutions in the population should be chosen from the infeasible solutions in \( IC \). To select \( N-|FC| \) infeasible solutions from \( IC \), we first modify the objective function values of these infeasible solutions. Because there are no feasible solutions in \( IC \), Equation (16) is modified as shown in Equation (18). In other words, the objective function values of the infeasible solutions in \( IC \) are revised in accordance with Equations (8), (11)-(15), (17), and (18).

\[ p_m(x, t) = v(x, t) \]

Subsequently, each individual in \( IC \) is assigned a fitness value that is calculated using Equation (10). Finally, we select \( N-|FC| \) infeasible solutions by means of the fast nondominated sorting and crowded-comparison operators proposed in [13].

b) The number of feasible solutions is more than \( NF \). When the number of feasible solutions is large, the number of feasible solutions that are propagated to the next generation may also be large. In this case, we propose to modify the objective function values of all solutions in \( C \) in accordance with Equations (11)-(17). Once the modified objective function values of all solutions have been obtained, \( N \) solutions are chosen by means of the fast nondominated sorting and crowded-comparison
operators proposed in [13].

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

<table>
<thead>
<tr>
<th>Algorithm 4 Population selection procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Input: ( P, Q, C=\Phi, FC=\Phi, IC=\Phi )</td>
</tr>
<tr>
<td>2: Output: ( P )</td>
</tr>
<tr>
<td>3: Cope all solutions in ( P ) and ( Q ) to ( C ).</td>
</tr>
<tr>
<td>4: Calculate the number of feasible solutions in ( P ) and ( Q );</td>
</tr>
<tr>
<td>5: if The number of feasible solutions ( \leq NF ) then</td>
</tr>
<tr>
<td>6: All feasible solutions in ( C ) are assigned to ( FC ), and infeasible ones are allocated to ( IC ).</td>
</tr>
<tr>
<td>7: Modify the objective function values of solutions in ( IC ) using Equations (8), (11)-(15), (17), and (18).</td>
</tr>
<tr>
<td>8: Calculate the fitness value of each solution using Equation (10).</td>
</tr>
<tr>
<td>9: Choose ( N/</td>
</tr>
<tr>
<td>10: Let ( P'=\Phi ), and cope all feasible solutions in ( FC ) and the selected ( N/</td>
</tr>
<tr>
<td>11: else</td>
</tr>
<tr>
<td>12: Modify the objective function values of all solutions in ( C ) according to Equations (11)-(17).</td>
</tr>
<tr>
<td>13: Calculate the fitness value of each solution, and choose ( N ) solutions from ( C ).</td>
</tr>
<tr>
<td>14: Let ( P'=\Phi ), and cope the selected ( N ) solutions to ( P ).</td>
</tr>
<tr>
<td>15: end if</td>
</tr>
</tbody>
</table>

E. Dynamism Handling

If the optimizer is aware of the changes in a DCMOP, then it does not need to detect change; otherwise, change detection is usually needed. When a change arises, change response has an important role in tracking the new POF in a changing environment. In this section, we first discuss the change detection method that detects the changes arising in DCMOPs with unforeseen environmental changes. Subsequently, the proposed change response strategy is described.

1) Change Detection

The authors in [11], [16], [17], and [18] proposed that changes can be detected by comparing previous objective function values with re-evaluated ones. However, this change detection method may also fail when only the constraints are dynamic. For a DCMOP, an environmental change may arise in either the objective functions or constraints. In this paper, we propose that both the objective functions and constraints are detected to accurately detect environmental changes. However, detection will be computationally expensive if each individual in the population is chosen as a detector. To improve detection efficiency, the number of individuals chosen as detectors is restricted to 10\% \( N \), which is suggested by [11] and [18]. The steps to detect environmental changes are as follows:

Step 1: Randomly choose 10\% \( N \) individuals from the population.

Step 2: Let \( t=1 \);

Step 3: If \( i \leq 10\%N \), then recalculate each objective function value for the \( t \)th solution; otherwise, the detection procedure ends.

Step 4: If any of the recalculated objective function values of the \( t \)th solution is different from their corresponding previous value, then a change is successfully detected, and the detection procedure ends; otherwise, go to Step 5.

Step 5: Recalculate the difference between the left- and right-hand sides of each constraint. If this value differs from the one in the previous generation, then a change is assumed to be successfully detected, and the detection procedure ends; otherwise, let \( t=t+1 \) and go back to Step 3.

2) Change Response

For a DCMOP, the true POF or feasible region may change with \( t \). The nondominated solutions in the POF obtained by an algorithm at \( t-1 \) may have become dominated or infeasible. Similarly, the infeasible solutions at \( t-1 \) may have become feasible or nondominated. Additionally, a dynamic environment may cause the feasible region to become larger or smaller. Simply discarding all previous solutions and randomly reinitializing the population might be beneficial for population diversity, but more optimization time will be needed for the algorithm to converge. In contrast, reusing all previous solutions to search for new nondominated solutions may reduce the population diversity. For these reasons, this paper adopts random immigrants and memory schemes to reinitialize the population when a new environment arises.

Based on the above discussion, the population in a new environment will be composed of 50\% reinitialized solutions and 50\% previous ones. Note that the reinitialized solutions are generated using the method discussed in the Initialization section and are copied into an empty set \( R \). To choose 50\% of the previous solutions, all previous solutions are re-evaluated, and their modified objective function values are calculated using Equations (8), (9) and (11)-(15). Then, we select the desired number of feasible previous solutions from the population \( P \) by means of the fast nondominated sorting and crowding-comparison operators proposed in [13].

To enable the algorithm track the new POF quickly, once a new environment is detected, we exploit information collected from the reinitialized solutions and new approximation of the POF to update the retained previous solutions. Specifically, the reinitialized solutions and retained solutions are first combined, and the feasible solutions are then copied into a new set \( FS \). Each solution in \( FS \) is assigned a fitness value. Considering that solutions with fitness values of less than two may be close to the new POF, we propose that each modified solution should include the information of these appealing solutions. For the \( t \)th retained previous solution, Equation (19) is used to locate a new position for each of its variables.

\[
x_{ij} = x_{ij} + \text{rand} \times (x_{\text{best},ij} - x_{ij})
\]

where \( x_{\text{best},ij} \) is the \( j \)th variable of a solution with a fitness value less than the fitness values of two randomly solutions from \( FS \).

However, the variable value obtained via Equation (19) may be smaller or greater than the corresponding lower or upper bound on that variable, making this variable infeasible. The bound constraints on each decision variable are known, so such infeasibilities can be repaired in a timely manner. Considering that the feasible region may become smaller or larger with dynamism, the feasibility ratio is considered in the repair method. To quickly calculate the feasibility ratio, we use a rule of thumb (i.e., calculating the percentage of feasible solutions during the reinitialization stage) to estimate the feasibility ratio at \( t \). The calculation method is given in Equation (20).

\[
\rho = \frac{NR}{N/2}
\]

where \( NR \) denotes the number of feasible solutions in the
reinitialization stage.

For an infeasible variable that exceeds its lower bound, its optimal value may be near to the corresponding lower bound. Therefore, we find a feasible variable that is near its lower bound. In contrast, a feasible variable close to its upper bound is optimal value may be near to the corresponding lower bound.

Algorithm 5 Change response

1: Input: $P$, $CSC=\Phi$, $R=\Phi$
2: Output: $P$
3: Generate $N/2$ solutions by the method in Section IV-A, and copy them into $R$;
4: Calculate the feasibility ratio of solutions in $R$ using Equation (20);
5: Calculate the modified objective values of all solutions in $P$;
6: Calculate the fitness values and crowding distances of all solutions in $P$;
7: Select 50% feasible old solutions from population $P$ according to the fitness values and crowding distances, and put them into a new set $CS$;
8: $CSC=CS\cup R$
9: Calculate the constraint violation values of solutions in CSC;
10: for $r=1$ to $|F|$
11: Random select an individual with fitness value less than two from $P$;
12: for $j=1$ to $n$
13: Generate $j$th variable $x_j$ by Equation (19);
14: if $x_j > x_j^{\text{min}}$ then
15: Repair it by Equations (21) and (23);
16: else
17: if $x_j < x_j^{\text{max}}$ then
18: Repair it by Equations (22) and (23);
19: end if
20: end if
21: end for
22: end for

The pseudocode for the change response procedure is presented in Algorithm 5.

F. Computational Complexity of One Generation of dCMOEA

In the loop (lines 6–15 of Algorithm 1) of each generation, computational resources are mainly consumed by the mating selection, offspring reproduction, population selection, and nondominated solution selection procedures, while other procedures require less computational cost. Selecting parents (i.e., the mating selection operator in line 11 of Algorithm 1) takes $O(MN)$ computations, where $M$ is the number of objectives. Generating an offspring solution (line 12 of Algorithm 1) requires $O(M)$ computations, so the offspring reproduction takes $O(MN)$ computations. The population selection procedure (line 13 of Algorithm 1) requires $O(MN^2)$ computations on the fitness assignment and $O(N^3 \log N)$ computations on elitist preservation. The nondominated solution selection procedure (line 15 of Algorithm 1) spends $O(MN^2)$ computations. Thus, the overall computational complexity of dCMOEA for one generation is $O(MN^2)$ or $O(N^3 \log N)$, whichever is larger.

V. EXPERIMENTAL STUDY

In this section, we will examine the effectiveness of the proposed test instances and dCMOEA. For the test instances, if the performances of several representative algorithms are clearly distinct, then this will show that the proposed test instances are effective. For dCMOEA, if it can outperform all algorithms considered for comparison on most or even some of the test instances, then this will suggest that dCMOEA is competitive for solving DCMOPs.

A. Compared Algorithms

Only a few MOEAs have already been proposed for solving DCMOPs. Therefore, in addition to two algorithms for solving DCMOPs (i.e., DC-MOEA, proposed in [24], and DNSGA-II-A, proposed in [25]), this paper also considers three representative constrained multiobjective optimization algorithms for solving CMOPs (i.e., constrained NSGA-II (C-NSGA-II) [15], constrained NSGA-III (C-NSGA-III) [31], and constrained two-archive EA (C-TAEA) [21]) for comparison. For handling dynamic environments, the nondominated solution selection operator and the change detection method proposed in this study and the restart scheme for change response were incorporated into C-NSGA-II, C-NSGA-III, and C-TAEA. The resulting algorithms are called DC-NSGA-II, DC-NSGA-III, and DC-TAEA, respectively.

B. Parameter Settings

1) To study the impact of the change frequency ($\tau_i$), $\tau_i$ was set to values of 10, 15, and 20. The number of changes was $n_c=21$. To minimize the effect of static optimization, each algorithm is allowed to run for 40 generations before the first change, and the total number of generations was $n_c \times \tau_i+40$.

2) Settings of the Test Problems: According to [6], some of the parameters used in Equation (2) were set as follows: $\theta=-0.25\pi$, $a=0.2$, $b=1$, $c=1$, $d=0.5$, and $e=1$. To cause the true POFs of the test problems to change in accordance with the characteristics of problems of Types I–IV, $\theta$, $A_i$, and $x(t)$ were set to $-\pi/16$, 0.05, and 6, respectively, and $x(t)$ and $m(t)$ were set as follows:

Type I: $x(t) = \max(3.5-0.14t, 0.7+0.14t)$
$m(t) = \max(1.43-0.05t, 0.43+0.05t)$

Type II: $x(t) = \max(2.5-0.05t, 1.5+0.05t)$
$m(t) = \max(1.16-0.075t, -0.34+0.075t)$

Type III: $x(t) = \min(2.1+0.14t, 4.9-0.14t)$
$m(t) = \min(0.93+0.05t, 1.93-0.05t)$

Type IV: $x(t) = \min(2+0.05t, 3-0.05t)$
$m(t) = \min(0.41+0.075t, 1.91-0.075t)$

In the designed test problems, $h(x_i, t)$ is set to $x_i$, and $g(x_i, t)$ is denoted as follows:

$g(x_i, t) = \sum_{j=2}^{d} (x_j - (1-0.9 \sin(0.2 \pi t)))^2$ (24)

To cause the objective functions to change in accordance with the modes corresponding to Cases 1 and 2, $W_i$ was set as follows:

Case 1: $W_i=2$;
Case 2: \( W_f = 6 \times \sin(0.2 \times \pi (t+1)) \).

Base on the above settings, eight test problems can be obtained, which are shown in Table I. The true POF and POS of each problem are given in Section S-IV of the supplementary material.

3) **Decision Variables.** Each test problem has ten decision variables; \( x_1 \in [0, 1] \) and \( x_i \in [0, 2] \) for \( i \geq 2 \).

4) **Algorithm Parameters:** Both the parent population size \( N_p \) and the offspring population size in all algorithms were set to 200. \( NF \) was set to 100. In SBX, the crossover probability \( (C_r) \) and the distribution index \( (de) \) were set to 0.8 and 5, respectively. In PM, the mutation probability \( (mp) \) and the distribution index \( (dm) \) were set to 0.05 and 40, respectively. The detailed experiments that select these parameters are given in Section S-V of the supplementary material.

The other parameters for the five compared algorithms were the same as those used in the referenced papers.

### C. Performance Indicators

In our experimental studies, three performance metrics, i.e., the hypervolume (HV) [8], inverted generational distance (IGD) [9], and Schott’s spacing metric (SP) [52] were adopted to compare the algorithms’ performance. The HV and IGD metrics simultaneously measure the diversity and convergence of the results, and the SP metric measures how the solutions in the discovered POF are distributed. Let \( POF^* \) be the obtained approximation set of POF.

1) **HV:** HV can assess the size of the area covered by the obtained approximation set. The reference point for the calculation of HV is set to \( (z_1+1, z_2+1) \), where \( z_1 \) and \( z_2 \) are the maximum values of two objective values of the true POF. A higher HV value means a better approximation set.

2) **IGD:** IGD is calculated by Equation (25).

\[
IGD = \frac{1}{N_{POF}} \sum_{i=1}^{N_{POF}} d_i
\]

where \( N_{POF} \) is the size of \( POF \), and \( d_i \) is the minimum Euclidean distance between the \( i \)th member in POF and one in \( POF^* \). The lower the IGD value, the better the obtained approximation set.

3) **SP:** Schott’s SP metric tests the distribution of the obtained POF, and SP is calculated by Equation (26). Note that a smaller SP value indicates a better quality of the results.

\[
SP = \frac{1}{N_{POF}} \sum_{i=1}^{N_{POF}} (D_i - \bar{D})^2
\]

where \( D_i \) is the Euclidean distance between the \( i \)th solution and its nearest member in \( POF^* \). \( \bar{D} \) is the average value of \( D_i \).

### D. Empirical Results

For each combination \((n, \tau)\) for a test problem, each algorithm was run 30 times on each test instance, and the mean and standard deviation of the results were recorded. Note that the Wilcoxon rank-sum test [53] at the 0.05 significance level was used to determine whether the performance of one algorithm statistically differs from that of another algorithm with respect to each performance metric.

To judge whether the proposed test instances are effective, the ranking method described in [4] was used. Specifically, an algorithm ranks the highest if it outperforms the largest number of competitors, and the algorithm that outperforms the fewest competitors will be assigned the worst rank. Multiple algorithms may have the same rank if they outperform the same number of other algorithms. Note that for each algorithm, the average rank was calculated based on three combinations for a given test problem. An algorithm was considered the best if it had the smallest average rank. Conversely, an algorithm was considered the worst if it had the largest average rank. Based on this ranking method, the obtained rank of each algorithm is listed in Section S-VI of the supplementary material.

It can be observed that \( dCMOEAV \) achieves the best performance in terms of all three metrics, followed by DC-NSGA-II, DC-NSGA-II-A performs the worst among the compared algorithms. DC-NSGA-III, DC-TAEA, and DC-MOEAV show different performances on different test problems. The experimental results indicate that the designed test problems can clearly distinguish the performance of each algorithm. Therefore, the proposed test problems are effective. To analyze the algorithms’ performance in terms of each of the three metrics in detail, the obtained average results and standard deviations for the HV, IGD, and SP metrics are shown in Tables II-IV, respectively. The computational time of each algorithm on each test problem is given in Section S-VII of the supplementary material.

As seen in Table III, on the proposed test instances, all five compared algorithms are outperformed by \( dCMOEAV \) in terms of the IGD metric. The results obtained by DC-NSGA-II are second only to those obtained by \( dCMOEAV \), whereas DNSGA-II-A performs the worst among the compared algorithms. Overall, the experimental results demonstrate the capability of \( dCMOEAV \) for solving DC-MOPs. As a supplement to the tabular presentation, Fig. 4 shows the evolutionary curves of the average IGD values on the first two test problems with \( \tau = 15 \) and \( n = 21 \). The evolutionary curves on the other test problems are given in Section S-VIII of the supplementary material. Note that the evolutionary curves of the results obtained by DC-NSGA-II-A are not included because of its poor performance. Compared with the other algorithms, \( dCMOEAV \) responds to environmental changes more steadily and recovers faster for the most of the test problems, implying its higher convergence performance. Nevertheless, \( dCMOEAV \) performs similarly to DC-NSGA-II but better than DC-MOEAV, DC-NSGA-III, and DC-TAEA on all test problems.

Table IV presents the results achieved by the six algorithms in terms of the SP metric. This table shows that \( dCMOEAV \) achieves the best results on most of the test problems. For the
IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION

TABLE II

<table>
<thead>
<tr>
<th>n</th>
<th>IGD (Mean ± Standard Deviation)</th>
<th>HV (Mean ± Standard Deviation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0648(0.0004)</td>
<td>0.0269(0.0001)</td>
</tr>
<tr>
<td>15</td>
<td>0.1236(0.0009)</td>
<td>0.0379(0.0002)</td>
</tr>
<tr>
<td>20</td>
<td>0.2080(0.0007)</td>
<td>0.0579(0.0001)</td>
</tr>
<tr>
<td>30</td>
<td>0.3207(0.0009)</td>
<td>0.1177(0.0001)</td>
</tr>
<tr>
<td>50</td>
<td>0.5648(0.0009)</td>
<td>0.2690(0.0002)</td>
</tr>
<tr>
<td>100</td>
<td>0.9242(0.0011)</td>
<td>0.5560(0.0002)</td>
</tr>
</tbody>
</table>

1 and * indicate dCMOEA performs significantly better than and comparable to the corresponding algorithm, respectively.

VI. DISCUSSION

A. Effectiveness of Each Component of dCMOEA

This subsection investigates the effects of different components of dCMOEA, including three key components, i.e., the mating selection operator, the population selection operator, and the change response strategy. To examine the effectiveness of each component of dCMOEA, we adapted the original dCMOEA into six variants. The first variant (dCMOEA-S1) uses the mating selection operator designed in [31] to replace the one designed in this paper. All variants were replaced with that proposed in [31].

second and sixth test problems with n=21 and τ=15 and the fifth test problem with n=21 and τ=20, however, DC-NSGA-II shows a performance similar to that of dCMOEA. This shows that DC-NSGA-II is promising for problems with slow changes, which may be because DC-NSGA-II has a slow convergence capability. For all test problems, DC-MOEA, DC-NSGA-II-A, DC-NSGA-II, DC-TAEA and DC-MOEA fail to show slow changes, which may be because DC-NSGA-II has a slow convergence capability.
DCMOPs. Notably, for nondominated solutions with a good distribution for generated. Therefore, the poor performance of solutions in the parent population, then more poor children are solutions are far from the true POF. If there are many such chosen for generating offspring. However, many feasible compared variants on all test problems, which may be because parent population.

As seen from Table V, dCMOEA performs significantly better than its variants on most of the test problems in terms of all three metrics, implying that these three key components are crucial for improving the performance of dCMOEA on DCMOPs. The IGD values of dCMOEA on the last four test problems are small, illustrating that dCMOEA has a better convergence capability for DCMOPs with oscillating objective function modes. Notably, dCMOEA-S3 shows similar performance in terms of the IG metric on the first four test and the eighth problems, mainly because dCMOEA-S3 and dCMOEA adopt the similar constraint handling techniques that are from [15]. In addition, dCMOEA achieves quite small SP values on all test problems, implying that dCMOEA can search for nondominated solutions with a good distribution for DCMOPs. Notably, dCMOEA-S5 shows similar performance in terms of the SP metric on the sixth test problem, mainly due to the diversity of the randomly generated solutions.

In dCMOEA-S1, feasible solutions can be more easily chosen for generating offspring. However, many feasible solutions are far from the true POF. If there are many such solutions in the parent population, then more poor children are generated. Therefore, the poor performance of dCMOEA-S1 may be due to a large number of poor feasible solutions in the parent population.

dCMOEA-S2 achieves the worst performance among the compared variants on all test problems, which may be because infeasible solutions are not considered if there are sufficient feasible solutions. Indeed, some infeasible solutions close to the true POF may carry promising population information and generate competitive offsprings. Simply abandoning them may reduce the population diversity. The poor performance of dCMOEA-S3 on some test problems may be attributable to the fact that fewer feasible solutions are retained when the feasible region is small. Note that the solutions in the POF must be feasible. Fewer feasible solutions reduce the number of nondominated solutions in the POF, decreasing the HV values and increasing the IGD and SP values.

dCMOEA-S4 does not consider any information on the new POF when an environmental change is detected, resulting in a poor initial population in the search for nondominated solutions. By contrast, when a change arises, dCMOEA-S5 completely reinitializes the population. For a DCMOP, some previously dominated solutions may become nondominated solutions in the new environment. The initial population obtained by dCMOEA-S5 has good diversity, but it miss many promising previous solutions. In dCMOEA-S6, the obtained solutions are not updated. However, the previous feasible region may become infeasible in the new environment, thus increasing the number of infeasible solutions. Therefore, dCMOEA-S6 fails to track the new POF.

In summary, by combining the three key components proposed in dCMOEA, dCMOEA outperforms all compared variants, showing that each component of dCMOEA plays an
important role in dealing with DCMOPs. Here, we would like to explain the role of each key component in more detail. The mating selection operator chooses the parents in accordance with the modified objective functions, and infeasible solutions are given a high probability of selection, thereby improving the diversity of the parents and allowing dCMOEA to utilize infeasible solutions efficiently and effectively to generate offspring with appealing performance. The population selection strategy can balance the convergence rate of the population with the population diversity. This is because when the number of feasible solutions becomes small, dCMOEA can retain all feasible solutions, thus helping to rapidly track the new POF and POS, whereas when the number of feasible solutions is large, infeasible solutions with small constraint violations and objective function values are propagated to the next generation, thereby diversifying the algorithm. Introducing random solutions in the change response stage can allow the algorithm to explore more information on the new environment to search for members of the population close to the new POF, and the update strategy for the retained previous solutions is beneficial for convergence when the true POF moves. In summary, these three key components of dCMOEA play important roles in solving DCMOPs.

**B. Influence of the Number of Decision Variables**

Our preliminary experimental results show that the computation time needed to obtain a feasible solution increases as the number of decision variables increases. Indeed, for the test problems with the current parameters, the computation time is unbalanced when the number of decision variables is more than 15. To examine the performance of dCMOEA on instances with 15 and 20 decision variables, we revised $t_i$, which controls the upper bound of the feasible region for the desired instance. For test problems with 15 decision variables, $t_i$ was revised to 8. The value of $t_i$ in test instances with 20 decision variables was revised to 12. Considering that DNSGA-II, DC-MOA, and DC-TAE are relatively weak in the above experiments, for the experiments reported in this subsection, the test problems with 15 and 20 decision variables were solved using only DC-NSGA-II, DC-NSGA-III, and dCMOEA. The values and standard deviations of the three performance metrics (i.e., HV, IG, and MS) for these algorithms on the test problems with 15 and 20 decision variables are listed in Section S-X of the supplementary material.

It can be observed from the corresponding tables that dCMOEA achieves the best performance on these instances among the three tested algorithms, thus demonstrating that dCMOEA can successfully solve DCMOPs with up to 20 decision variables.

**VII. Conclusion**

In a DCMOP, the true POF is determined by the unconstrained POF and feasible region. This paper proposes a set of test instances for DCMOPs that consider simultaneous changes in the feasible region and unconstrained POF. The unconstrained POFs in the proposed test problems exhibit two modes. In one mode, their shapes remain unchanged with dynamism; in the other mode, the shapes of POFs oscillate among several optimization modes. During optimization, the feasible region may increase or decrease with dynamism, causing the true POF to become disconnected. The proposed test problems can be used to judge an algorithm's capability of tracking the new POF and handling infeasible solutions. Furthermore, six dynamic constrained MOEAs for optimization were tested on eight test problems, and the results were evaluated in terms of three performance metrics. The comparison shows that the designed instances are effective and useful for distinguishing the performance of each algorithm.
reuses some previous solutions with small fitness values and re-evaluates them. Afterward, dCMOEA updates these previous solutions on the basis of useful information obtained from the new environment, i.e., the feasibility ratio and the newly obtained nondominated solutions, to provide a reinitialized population for the algorithm.

Despite that dCMOEA has shown appealing performance on the proposed test problems, this paper has several aspects that limit its applicability in certain situations. The parameter settings for $s(t)$, $m(t)$, and $z(t)$ used in the proposed benchmark problems depend heavily on trial and error. Moreover, this study does not use dCMOEA to solve the DCMOPs with three or many objectives. Regarding the change response, dCMOEA is designed for tracking the new true POF with no consideration about the transition cost of a new solution.

Considering that scalability plays a crucial role in designing an efficient algorithm [55], in the future we will study the scalable dynamic constrained many-objective optimization algorithms and design the benchmark problems that are close to real-world applications and can easily be extended to three or more objective functions. In many real-world applications, changing the production solution introduces additional cost [64]. Thus, we will borrow the ideas proposed in [57] to design a general framework that can find robust solutions. In addition, there are other types of changes in the true POF and the feasible region of objective functions, which we will study in the future.

REFERENCES


Qingda Chen (S’19) received the B.S. degree from Yantai University, Yantai, China, in 2013. Since 2014, he has been working toward the Ph.D. degree in control theory and control engineering at Northeastern University. His current research interests include computational intelligence and its application in the industrial processes.

Jinliang Ding (SM’14) received the Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2012. His current research interests include modeling, plant-wide control and optimization for the complex industrial systems, stochastic distribution control, and multiobjective evolutionary algorithms and its application.

Prof. Ding was a recipient of the Young Scholars Science and Technology Award of China in 2016, the National Science Fund for Distinguished Young Scholars in 2015 and the T1 National Technological Invention Award in 2013.

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, where he became a Professor 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 International Eurasian Academy of Sciences, and IFAC Fellow.
A Novel Evolutionary Algorithm for Dynamic Constrained Multiobjective Optimization Problems

—Supplementary Material

Qingda Chen, Student Member, IEEE, Jinliang Ding, Senior Member, IEEE, Shengxiang Yang, Senior Member, IEEE, and Tianyou Chai, Fellow, IEEE

This is the supplementary material to the paper entitled “A Novel Evolutionary Algorithm for Dynamic Constrained Multiobjective Optimization Problems”, submitted to IEEE Transactions on Evolutionary Computation.

CONTENTS
Section S-I Mathematical functions described in Fig. 1.
Section S-II Real-world applications linking to the four types of the proposed problems.
Section S-III Ideas on scaling up the proposed test problems
Section S-IV True POF and POS of each test problem
Section S-V Algorithm parameter selection
Section S-VI Performance ranking of each algorithm.
Section S-VII Computational time of each algorithm on each test problem.
Section S-VIII Evolution curves of average IGD values for the third to eighth test problems.
Section S-IX Three performance metrics of algorithms on the test problems with 15 and 20 decision variables.

S-I. MATHEMATICAL FUNCTIONS DESCRIBED IN FIG. 1
In this paper, the true POF of a dynamic constrained multiobjective optimization problem (DCMOP) is given in Fig. 1 of this paper, and the corresponding mathematical functions of this DCMOP are Equation (S-1) in which \( t \) is set to 7.

\[
\begin{align*}
\min F(x,t) &= (f_1(x,t)+f_2(x,t))^2 \\
\text{obj:} \\
&f_1(x,t) = (1 + \sum_{j=1}^{n} (x_j - \sin(0.05\pi x_j))^2)(x_1 + 0.05 \sin(4\pi x_1)) \\
&f_2(x,t) = (1 + \sum_{j=1}^{n} (x_j - \sin(0.05\pi x_j))^2)(3 - 0.05 x_1 - 0.05 \sin(4\pi x_1)) \\
&f_3(x,t) = \sin(\frac{x_3}{4}) \times f_1(x,t) - 2.36 + 0.01t \geq 0 \\
&s.t.: \quad 0.2 \times \sin(4\pi x \times (\sin(\frac{\pi}{16}) \times f_1(x,t) - 1) + \cos(\frac{\pi}{16}) \times f_2(x,t)) \leq 0 \\
&0.73 \times f_1(x,t) + f_2(x,t) - 6 < 0
\end{align*}
\]

S-II. REAL-WORLD APPLICATIONS LINKING TO THE FOUR TYPES OF THE PROPOSED PROBLEMS
The real-world multiobjective optimization problems (MOPs) involved in the existing literature are mainly MOPs with constraints or dynamism. To the best of our knowledge, the real-world optimization problems solved in the existing literature do not consider dynamism, constraints, and multiobjective simultaneously. Indeed, many real-world problems are DCMOPs. Considering this fact, we take the fluid catalytic cracking-distillation (FCC-D) process, a DCMOP we are working on, as an example to illustrate the four types of the true POF and the feasible region mentioned in Section III of this paper.

The FCC-D process is one of the most energy-consuming steps in refineries, which involves complicated physical and chemical reactions. In the FCC-D process, the heavy oil is cracked into the light hydrocarbons in the presence of catalyst, and then the light hydrocarbons are separated into different products (i.e., gasoline, diesel, and natural gas) by the distillation process. To minimize the energy consumption and maximize the economic benefits of products, refineries usually optimize the operating variables that control the stable operation of production units according to production constraints (i.e., yield constraints of different products). During
the optimization of the operating variables, the operator usually changes the yield constraints of products to avoid large production fluctuations and production accidents when the production scenarios are switched. Therefore, optimizing the operating variables of the FCC-D process is a DCMOP, which aims at minimizing energy consumption and maximizing the economic benefits of the products, and takes product yields as dynamic constraints.

For the ease of readers understanding, we will describe the mathematical model of the FCC-D process in Equation (S-2). The definitions of the indices, variables and parameters employed in the model are given as follows:

**Indices**

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>Product index, $p=1$ (natural gas), 2 (gasoline), and 3 (diesel).</td>
</tr>
<tr>
<td>$t$</td>
<td>Dynamic environment index.</td>
</tr>
<tr>
<td>$w$</td>
<td>Pumparound (PA) index.</td>
</tr>
</tbody>
</table>

**Variables**

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X(t)$</td>
<td>Operating variable vector at $t$.</td>
</tr>
<tr>
<td>$x_j(t)$</td>
<td>The $j$th operating variable at $t$.</td>
</tr>
<tr>
<td>$Y_p(t)$</td>
<td>The $p$th product yield at $t$.</td>
</tr>
<tr>
<td>$YP(t)$</td>
<td>Yield of pollutant at $t$.</td>
</tr>
<tr>
<td>$HF(t)$</td>
<td>Mass flow rate of heavy oil at $t$.</td>
</tr>
<tr>
<td>$CT(t)$</td>
<td>Temperature of the heated catalyst at $t$.</td>
</tr>
<tr>
<td>$RT(t)$</td>
<td>Outlet temperature of the riser at $t$.</td>
</tr>
<tr>
<td>$CF(t)$</td>
<td>Mass flow rate of the catalyst circulation rate at $t$.</td>
</tr>
<tr>
<td>$SF(t)$</td>
<td>Flow rate of the stripping steam at $t$.</td>
</tr>
<tr>
<td>$LF(t)$</td>
<td>Flow rate of the lifting steam at $t$.</td>
</tr>
<tr>
<td>$PQ_j(t)$</td>
<td>Calorific value of heat recovery in the $j$th PA of the fractionator at $t$.</td>
</tr>
<tr>
<td>$HT(t)$</td>
<td>Temperature of the heavy oil at $t$.</td>
</tr>
<tr>
<td>$Ylp(t)$</td>
<td>Lower bound of demand for the $p$th product at $t$.</td>
</tr>
<tr>
<td>$Yup(t)$</td>
<td>Upper bound of demand for the $p$th product at $t$.</td>
</tr>
</tbody>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PY_p$</td>
<td>Unit price of the $p$th product.</td>
</tr>
<tr>
<td>$cYS$</td>
<td>Unit price of removing pollutant.</td>
</tr>
<tr>
<td>$cSS$</td>
<td>Unit price of heavy oil.</td>
</tr>
<tr>
<td>$cFF$</td>
<td>Unit price of steam.</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Heat transfer efficiency in the PAs.</td>
</tr>
<tr>
<td>$FT$</td>
<td>Energy required to raise 1 mol of heavy oil by 1 °C.</td>
</tr>
<tr>
<td>$I_{x_j}$</td>
<td>Lower bound of $x_j(t)$.</td>
</tr>
<tr>
<td>$u_{x_j}$</td>
<td>Upper bound of $x_j(t)$.</td>
</tr>
</tbody>
</table>

The mathematical model of the FCC-D process can be described as follows [i.e., Equation (S-2)]:

$$
\begin{align*}
\max f_{\text{max}}(X(t)) &= \sum_{p} (YP_p(t) - cSS \times YP(t) - cSS \times HF(t) - CT(t) \times CF(t) - SF(t) \times LF(t) - HF(t) \times RT(t) - HT(t)) \\
\min f_{\text{min}}(X(t)) &= 0, \quad k = 1, \ldots, h \\
&\text{subject to} \\
&I_{x_j}(t) \leq x_j(t) \leq u_{x_j}(t) \\
&Ylp(t) \leq Y_p(t) \leq Yup(t) \\
&Ylp(t) = g_p(X(t)) \\
&[Y_p(t) = \min \{CF(t), CT(t), SF(t), LF(t), HF(t), RT(t), HT(t)\}] \\
&[x_j(t) \in \{CF(t), CT(t), SF(t), LF(t), HF(t), RT(t), HT(t), PQ_1(t), PQ_2(t), PQ_3(t)\}] \\
&x_j(t) \in \{CF(t), CT(t), SF(t), LF(t), HF(t), RT(t), HT(t), PQ_1(t), PQ_2(t), PQ_3(t)\}
\end{align*}
$$

(S-2)

where $h$ is the number of equality constraints, and $g_p(X(t))$ is the $k$th equality constraint. $g_p(X(t))$ denotes the complicated nonlinear relationship between the $p$th product yield and $X(t)$. $g_p(X(t))$ models either by using neural networks (e.g., reference [s1]) or by using [s2] as a rigorous model.

This paper uses $t=1$, 2, and 3 to denote different dynamic environments of the DCMOP in the FCC-D process (FCC-D-DCMOP). Subsequently, we provides the diagrams of true POF and feasible regions arising in the four types mentioned in Section III of this paper. Specifically, $t=1$ denotes the current environment of the FCC-D-DCMOP, $t=2$ represents the environment in which product yield constraints change due to different production scenarios, and $t=3$ denotes the environment in which the operator adjusts the product yield constraints based on the optimized operating variables.

Note that the real POF and the feasible region of the FCC-D-DCMOP, in practice, show hundreds of types that vary with dynamic environments, and we only describe four typical types corresponding to the test problems proposed in this paper.

1. **Type I**

The feasible region of the FCC-D-DCMOP becomes small if the feasible ranges of product yields are small at $t=1$ (e.g., $Yu_p(t)$ decreases, and $Yl_p(t)$ increases at $t=1$), which may lead to the infeasibility of the unconstrained POF of the FCC-D-DCMOP (i.e., the unconstrained POF is not in the feasible region) and cause that the true POF of the FCC-D-DCMOP is only determined by the lower bound of the feasible region. In practice, the lower bound of the feasible region may be continuous if the solutions (i.e., $X(t)$) corresponding to the lower bound of the feasible region are far from the boundary of $X(t)$, causing that the true POF of the FCC-D-DCMOP is continuous at $t=1$, as shown in Fig. S-1-a).

The feasible region of the FCC-D-DCMOP becomes large when the feasible ranges of product yields become large at $t=2$ (e.g., $Yu_p(t)$ increases, and $Yl_p(t)$ decreases at $t=2$), which may cause that the true POF of the FCC-D-DCMOP is determined by both the unconstrained POF and the lower bound of the feasible region. The true POF of the FCC-D-DCMOP may
change from continuous to disconnected if there are infeasible regions on the bound of the feasible region, as shown in Fig. S-1-b).

The operator needs to adjust the product yield constraints (i.e., \( t = 3 \)) if the optimized operating variables differ greatly from the current ones, which can avoid large production fluctuations and production accidents when the operating variables are switched. Narrowing the feasible range of product yields leads to the decrease of the feasible region and the movement of the true POF, which may cause that the true POF of the FCC-D-DCMOP is determined by the unconstrained POF and the lower bound of the feasible region. The true POF may return to continuous from disconnected when a part of the unconstrained POF coincides with the lower bound part of the feasible region, as shown in Fig. S-1-c).

It can be observed from Fig. S-1-a), Fig. S-1-b), and Fig. S-1-c) that the true POF and the feasible region accord with the characteristics discussed in Type I of this paper at \( t = 1 \), \( t = 2 \), and \( t = 3 \), respectively. (i.e., the feasible region first increases with time \( t \) and then decreases, while the true POF changes from continuous to disconnected and finally back to continuous).

2) Type II

The feasible region of the FCC-D-DCMOP becomes large if the feasible ranges of product yields are large at \( t = 1 \) (e.g., \( Y_{up}(t) \) increases, and \( Y_{lp}(t) \) decreases at \( t = 1 \)), which may cause that the true POF of the FCC-D-DCMOP is determined by both the unconstrained POF and the lower bound of the feasible region. The lower bound of the feasible region may be disconnected if there are infeasible regions in the lower bound of the feasible region, causing that the true POF of the FCC-D-DCMOP is disconnected at \( t = 1 \), as shown in Fig. S-1-d).

The feasible region of the FCC-D-DCMOP becomes small if the feasible ranges of product yields are small at \( t = 2 \) (e.g., \( Y_{up}(t) \) decreases, and \( Y_{lp}(t) \) increases at \( t = 2 \)), which may lead to the unconstrained POF of the FCC-D-DCMOP to be infeasible (i.e., the unconstrained POF is not in the feasible region) and cause that the true POF of the FCC-D-DCMOP is only determined by the lower bound of the feasible region. In practice, the lower bound of the feasible region may be continuous when the solutions (i.e, \( X(t) \)) corresponding to the lower bound of the feasible region are far from the bound of \( X(t) \), causing the true POF of the FCC-D-DCMOP to change from continuous to disconnected at \( t = 2 \), as shown in Fig. S-1-e).

The operator needs to adjust the product yield constraints (i.e., \( t = 3 \)) if the optimized operating variables differ greatly from the current ones, which can avoid large production fluctuations and production accidents when the operating variables are switched. Enlarging the feasible ranges of product yields can lead to the increase of the feasible region of objectives and the movement of the true POF, which may cause that the true POF of the FCC-D-DCMOP is determined by both
the unconstrained POF and the lower bound of the feasible region. The true POF may return to disconnected from continuous if there are infeasible regions on the bound of the feasible region, as shown in Fig. S-1-f).

It can be observed from Fig. S-1-d), Fig. S-1-e), and Fig. S-1-f) that the true POF and the feasible region accord with the characteristics discussed in Type II of this paper at $t=1$, $t=2$, and $t=3$, respectively. (i.e., the feasible region first decreases with time $t$ and then increases, while the true POF changes from disconnected to continuous and finally back to disconnected).

3) Type III

The feasible region of the FCC-D-DCMOP becomes small if the feasible ranges of product yields are small at $t=1$ (e.g., $Y_{rp}(t)$ decreases, and $Y_{lp}(t)$ increases at $t=1$), which may lead to the infeasibility of the unconstrained POF of the FCC-D-DCMOP (i.e., the unconstrained POF is not in the feasible region) and cause that the true POF of the FCC-D-DCMOP is only determined by the lower bound of the feasible region. In practice, the lower bound of the feasible region may be disconnected if the solutions (i.e, $X(t)$) corresponding to the lower bound of the feasible region are close to the bound of $X(t)$, causing the true POF of the FCC-D-DCMOP to be disconnected at $t=1$, as shown in Fig. S-1-g).

The feasible region of the FCC-D-DCMOP becomes large when the feasible ranges of product yields are large at $t=2$ (e.g., $Y_{rp}(t)$ increases, and $Y_{lp}(t)$ decreases at $t=2$), which may cause that the true POF of the FCC-D-DCMOP is determined by both the unconstrained POF and the lower bound of the feasible region. The true POF may return to continuous from disconnected if a part of the unconstrained POF coincides with the lower bound part of the feasible region, as shown in Fig. S-1-h).

The operator has to adjust the product yield constraints (i.e., $t=3$) if the optimized operating variables differ greatly from the current ones, which can avoid large production fluctuations and production accidents when the operating variables are switched. Narrowing the feasible range of product yields leads to the decrease of the feasible region and the movement of the true POF, which may cause that the true POF of the FCC-D-DCMOP is determined by both the unconstrained POF and the lower bound of the feasible region. The true POF may return to disconnected from continuous if some infeasible regions move to the bound of the feasible region, as shown in Fig. S-1-i).

It can be observed from Fig. S-1-g), Fig. S-1-h), and Fig. S-1-i) that the true POF and the feasible region accord with the characteristics discussed in Type III of this paper at $t=1$, $t=2$, and $t=3$, respectively. (i.e., the feasible region first increases with time $t$ and then decreases, while the true POF changes from disconnected to continuous and finally back to disconnected).
The true POF of the FCC-D-DCMOP is only determined by the unconstrained POF and cause that the true POF of the FCC-D-DCMOP is only determined by the unconstrained POF of the FCC-D-DCMOP to be infeasible (i.e., the unconstrained POF is not in the feasible region) and cause that the true POF of the FCC-D-DCMOP is only determined by the lower bound of the feasible region. In practice, the lower bound of the feasible region may be continuous if the solutions (i.e., X(t)) corresponding to the lower bound of the feasible region are far from the bound of X(t). Therefore, the true POF of the FCC-D-DCMOP is continuous at t=1 if a part of the unconstrained POF coincides with the lower bound part of the feasible region, as shown in Fig. S-1-j).

The feasible region of the FCC-D-DCMOP becomes small if the feasible ranges of product yields are small at t=1 (e.g., Ylp(t) decreases, and Ysp(t) increases at t=1), which may lead to the infeasibility of the unconstrained POF of the FCC-D-DCMOP (i.e., the unconstrained POF is not in the feasible region) and cause that the true POF of the FCC-D-DCMOP is only determined by the lower bound of the feasible region. In practice, the lower bound of the feasible region may be disconnected if the solutions (i.e, X(t)) corresponding to the lower bound of the feasible region are close to the bound of X(t), causing the true POF of the FCC-D-DCMOP to be disconnected at t=1, as shown in Fig. S-1-k).

The operator has to adjust the product yield constraints (i.e., t=3) if the optimized operating variables differ greatly from the current ones, which can avoid large production fluctuations and production accidents when the operating variables are switched. Enlarging the feasible range of product yields leads to the increase of the feasible region of objectives and the movement of the true POF, which may cause that the true POF of the FCC-D-DCMOP is determined by the unconstrained POF and the lower bound of the feasible region. The true POF may return to continuous from disconnected when a part of the unconstrained POF coincides with the lower bound part of the feasible region, as shown in Fig. S-1-l). It can be observed from Fig. S-1-j), Fig. S-1-k), and Fig. S-1-l) that the true POF and the feasible region accord with the characteristics discussed in Type IV of this paper at t=1, t=2, and t=3, respectively. (i.e., the feasible region first decreases with time t and then increases, while the true POF changes from continuous to disconnected and finally back to continuous).

Remark: Indeed, the true POF and the feasible region of the FCC-D-DCMOP show hundreds of types that vary with dynamic environments, and we only describe four typical types corresponding to the test problems proposed in this paper. In the future, other types of the true POF and the feasible region arising in real-world applications will be gradually explored.

S-III. IDEAS ON SCALING UP THE PROPOSED TEST PROBLEMS

The test problems proposed in Section III of this paper can be extended from the perspectives of the decision space and the objective space, as follows:

1) Decision Space

$h(x_I, t)$ and $g(x_{II}, t)$ in the proposed test problems determine the decision space, where $x_I \cap x_{II} = x$ and $x_I \cap x_{II} = \emptyset$. Indeed, the decision variables used in the proposed test problems can be extended in any dimension as long as the following conditions are satisfied:

a) The minimum value of $g(x_I, t)$ is zero.

b) $h(x_I, t)$ is set to $x_I$, and it satisfies $0 \leq h(x_I, t) \leq s(t)$.

Note that for the test problems designed in this paper, the real POF does not depend on the dimension of the decision
variables. Therefore, the real POF and the feasible region of the test problems can still change according to the four types mentioned in this paper as long as the settings of \( s(t), m(t), \) and \( \varepsilon(t) \) are reasonable. We recommend that their settings (i.e., \( s(t), m(t), \) and \( \varepsilon(t) \)) are consistent with those used in our paper when the dimensions of the test problem are extended.

2) Objective Function Space

In the proposed problems, the constraints cause the bound of the feasible region of the objectives to be irregular, leading to the true POF exhibiting different characteristics (i.e., disconnected and continuous) with dynamism. Therefore, we should first consider how to scale up the feasible regions of the objective functions.

In the proposed test problems, the irregularity of the bounds of the objective functions is caused by the dimensions of the test problem are extended. It can be ascertained from Section III of this paper that the unconstrained POF in at least one environment.

The true POF and POS of each test problem are given in Table S-I.

S-V. ALGORITHM PARAMETER SELECTION

In this paper, we adopt the popular simulated binary crossover (SBX) and polynomial mutation (PM) operators proposed in [54] to generate offspring. The main parameters in SBX and PM are the crossover probability (\( Cr \)), the distribution index for SBX (\( dc \)), the mutation probability (\( mp \)), and the distribution index for mutation (\( dm \)). Additionally, the other parameters in dCMOEA are the population size (\( N \)) and the threshold of the number of feasible solutions (\( Nf \)).

According to our preliminary experiments, a larger \( N \) leads to better performance of dCMOEA and a longer computation time. To eliminate the disturbance of \( N \) on the algorithms’ performance comparison, \( N \) of all algorithms was set to 200 (i.e., \( N=200 \)).

In Section IV-D, we proposed a population selection operator that updates the population (i.e., \( P \)) according to population and offspring (i.e., \( Q \)). The number of feasible solutions in \( P \) and \( Q \) of each generation usually is not the same (i.e., it may be zero, 100, 200, or 400). To determine \( Nf \), we set \( Nf \) to 0, 100, 200, 300, and 400 to analyze the sensitivity on \( Nf \) of dCMOEA. The results of sensitivity analysis on \( Nf \) are given in Table S-II based on the eight test problems.

It can be seen from Table S-II that dCMOEA with \( Nf=100 \) can perform better than its variants on most of the test problems in terms of three metrics, implying that \( Nf=100 \) is suitable for dCMOEA.

The parameters \( Cr \) and \( dc \) used in SBX are coupled to affect the performance of dCMOEA, and a similar situation for the other two parameters used in PM also arise. Therefore, \( Cr \) and \( dc \) are determined first, followed by \( mp \) and \( dm \). In this paper, we use three performance metrics (i.e., \( HV, IGD \), and \( SP \)) to evaluate the performance of an algorithm. For \( HV \), the larger the value, the better the performance of an algorithm. For \( IGD \) and \( SP \), however, the lower the value, the better the performance of an algorithm. To facilitate the analysis of the four parameters on the performance of dCMOEA, we use the average regularization indicator (ARI) to represent these three metrics, which is calculated as Equations (S-5) and (S-6).

\[
RI_{\text{bf}} = \frac{HV_{\text{bf}}}{HV_{\text{min}}} - \frac{IGD_{\text{bf}}}{IGD_{\text{min}}} - \frac{SP_{\text{bf}}}{SP_{\text{min}}} \tag{S-5}
\]

\[
ARI_{\text{bf}} = \frac{1}{8} \sum_{i=1}^{8} RI_{\text{bf}} \tag{S-6}
\]

where \( RI_{\text{bf}} \) is the regularization value of the performance metrics obtained by dCMOEA that uses the \( b \)th parameter combination to solve the \( f \)th test problem. \( HV_{\text{bf}}, IGD_{\text{bf}}, \) and \( SP_{\text{bf}} \) are, respectively, the \( HV, IGD \), and \( SP \) metrics obtained by dCMOEA based on the \( b \)th parameter combination to solve the \( f \)th test problem. \( ARIs \) denotes the sum of the average regularization indicators of dCMOEA that uses the \( b \)th parameter combination to solve eight test problems.

Table S-III provides the five levels of these four parameters based on our preliminary experiments. For example, \( Cr, dc, mp, \) and \( dm \) are, respectively, equal to 0.8, 7, 0.15, and 40 when their levels are 4. For brevity, an orthogonal array \( L_{25} \) [i.e., these two parameters have 25 (i.e., 5²) parameter combinations under five levels] was used to examine the first two parameters (i.e., \( Cr \) and \( dc \)). For each parameter combination (e.g., \( Cr=0.6 \) and \( dc=3 \)), dCMOEA was independently executed 30 times on
<table>
<thead>
<tr>
<th>Instance No.</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 1           | $s(t)=\max(3.5-0.14t, 0.7+0.14t); m(t)=\max(1.43-0.05t, 0.43+0.05t)$  
$W_{\text{opt}}=g(x_{\text{opt}})=\sum_{j=2}^{d}x_{j}(1-0.9xj \sin(0.2\pi xj))^{2}$  
$f(x, t) = (1+g(x_{j}_j)\mid x, 0.05 \sin(W_{\text{ratio}}))$  
$P_{\text{opt}}(x, t) = 0.55 \leqslant t, y = 1.0x \sin(0.2x), y_{j} = 2, \ldots, d_{j}$  
$POF(t) = f_{1} + f_{2} = s(t) + 2A_{1}\sin(W_{\frac{f_{1}+f_{2}}{2}})$  
$s_{1} \cos \frac{\pi}{2} x_{1} f_{1} - \sin \frac{\pi}{4} x_{1} f_{1} - m(t) \geq a \times \sin(x \times (\sin \frac{\pi}{16} (f_{1} - 1) + 0.05 x f_{1} ))^{1 \frac{1}{5}}$  
$f_{1} + f_{2} - 6 < 0; \ m(t) < 0$ | The unconstrained POF has one mode. The feasible region first increases with time $t$ and then decreases, while the true POF changes from continuous to disconnected and finally back to continuous. |
| 2           | $s(t)=\max(2.5-0.05t, 1.5+0.05t); m(t)=\max(1.16-0.075t, 0.34+0.075t)$  
$W_{\text{opt}}=g(x_{\text{opt}})=\sum_{j=2}^{d}x_{j}(1-0.9xj \sin(0.2\pi xj))^{2}$  
$f(x, t) = (1+g(x_{j}_j)\mid x, 0.05 \sin(W_{\text{ratio}}))$  
$P_{\text{opt}}(x, t) = 0.55 \leqslant t, y = 1.0x \sin(0.2x), y_{j} = 2, \ldots, d_{j}$  
$POF(t) = f_{1} + f_{2} = s(t) + 2A_{1}\sin(W_{\frac{f_{1}+f_{2}}{2}})$  
$s_{1} \cos \frac{\pi}{2} x_{1} f_{1} - \sin \frac{\pi}{4} x_{1} f_{1} - m(t) \geq a \times \sin(x \times (\sin \frac{\pi}{16} (f_{1} - 1) + 0.05 x f_{1} ))^{1 \frac{1}{5}}$  
$f_{1} + f_{2} - 6 < 0; \ m(t) < 0$ | The unconstrained POF has one mode. The feasible region first increases with time $t$ and then decreases, while the true POF changes from disconnected to continuous and finally back to disconnected. |
| 3           | $s(t)=\max(2.1-0.14t, 4.9+0.14t); m(t)=\max(1.43-0.05t, 0.43+0.05t)$  
$W_{\text{opt}}=g(x_{\text{opt}})=\sum_{j=2}^{d}x_{j}(1-0.9xj \sin(0.2\pi xj))^{2}$  
$f(x, t) = (1+g(x_{j}_j)\mid x, 0.05 \sin(W_{\text{ratio}}))$  
$P_{\text{opt}}(x, t) = 0.55 \leqslant t, y = 1.0x \sin(0.2x), y_{j} = 2, \ldots, d_{j}$  
$POF(t) = f_{1} + f_{2} = s(t) + 2A_{1}\sin(W_{\frac{f_{1}+f_{2}}{2}})$  
$s_{1} \cos \frac{\pi}{2} x_{1} f_{1} - \sin \frac{\pi}{4} x_{1} f_{1} - m(t) \geq a \times \sin(x \times (\sin \frac{\pi}{16} (f_{1} - 1) + 0.05 x f_{1} ))^{1 \frac{1}{5}}$  
$f_{1} + f_{2} - 6 < 0; \ m(t) < 0$ | The unconstrained POF has one mode. The feasible region first increases with time $t$ and then decreases, while the true POF changes from disconnected to continuous and finally back to disconnected. |
Thus, good exploration capability but poor exploitation capability. The exploration capabilities of level 4 and value of exploration capability. Similarly, a large value of factor-level trends of impacts of parameters on the performance of each test problem to obtain the corresponding ARI parameters Level 1 Level 2 Level 3 Level 4 Level 5

SENSITIVITY ANALYSIS ON DIFFERENT_THRESHOLDS OF NUMBER OF THE NUMBER OF FEASIBLE SOLUTIONS

It can be seen from Table S-III and Fig. S-2 that the method using to determine Cr and dc. For a similar reason, Crd and mp and dm are given in Fig. S-2. It can be seen from Table S-III and Fig. S-3 that Cr with a level of 4 and dc with a level of 3 yield the best result. In other words, mp and dm were set to 0.05 and 40, respectively.

S-VI. PERFORMANCE RANKING OF EACH ALGORITHM

The final rank of each algorithm under performance metrics for each test problem is indicated in Table S-IV.

S-VII. COMPUTATIONAL TIME OF EACH ALGORITHM ON EACH TEST PROBLEM

In Section V of this paper, we set the change frequency (τ) as 10, 15, and 20 (i.e., τ=10, 15, and 20), and τ=10 means that an algorithm iterates 10 generations in a dynamic environment. To further examine the performance of each algorithm, we compare the computational time of each algorithm in an environment. For a test problem with τ=10 and n=21, for example, the computational time means the time required for 10 iterations of an algorithm. We use Equation (S-7) to calculate the average computational time required for the pth algorithm to run for τ generations on the test problems.

$$CT_{p_i} = \frac{1}{r} \sum_{j=1}^{r} T_{p_{ij}}$$

where $T_{p_{ij}}$ denotes the time that is required to iterate τ times under the jth dynamic environment in the ith run of the pth algorithm. $CT_{p_i}$ denotes the average computational time required for the pth algorithm to run for τ generations. r is the number of runs of an algorithm on a test problem.

The average computational time of each algorithm on each test problem with τ=10, 15, and 20 is given in Table S-V. It can be seen from Table S-V that the computational time of dCMOEA is longer than that of DC-MOEA, DC-NSGA-II-A, DC-NSGA-II, and DC-NSGA-III on the most of the test problems.

To eliminate the influence of running time on the algorithm’s performance, we used an elapsed CPU time limit of CP=2000ms as the termination criterion of each algorithm in each dynamic environment. The average values and standard deviations of the three performance metrics (i.e., HV, IGD, and SP) for DC-MOEA, DC-NSGA-II-A, DC-NSGA-II,
DC-NSGA-III, and dCMOEA are given in Table S-VI. As seen from Table S-VI, dCMOEA performs significantly better than DC-MOEA, DC-NSGA-II-A, DC-NSGA-II, and DC-NSGA-III on most of the test problems in terms of the three metrics. Therefore, even with the same running time, the proposed algorithm still outperforms the others (i.e., DC-MOEA, DC-NSGA-II-A, DC-NSGA-II, and DC-NSGA-III).

S-VIII. EVOLUTION CURVES OF AVERAGE IGD VALUES FOR THE THIRD TO EIGHT TEST PROBLEMS

Fig. S-4 depicts the evolution curves of average IGD values for the third to eight test problems with $\tau=10$ and $n=21$.

S-IX. THE POFs OBTAINED BY THE COMPARED ALGORITHMS

Figures S-5–S-12 show the obtained POFs obtained by the compared algorithms on the eight test problems.

**TABLE S-IV**

<table>
<thead>
<tr>
<th>Instance No.</th>
<th>Rank</th>
<th>Ranking by HV</th>
<th>Ranking by IGD</th>
<th>Ranking by SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>DC-MOEA</td>
<td>DC-MOEA</td>
<td>DC-MOEA</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>DC-NSGA-II</td>
<td>DC-NSGA-II</td>
<td>DC-NSGA-II</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>DC-NSGA-III</td>
<td>DC-NSGA-III</td>
<td>DC-TAEA</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>DC-TAEA</td>
<td>DC-TAEA</td>
<td>DC-MOEA</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>DC-NSGA-II</td>
<td>DC-NSGA-II</td>
<td>DC-NSGA-III</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>DC-NSGA-II-A</td>
<td>DC-NSGA-II-A</td>
<td>DC-NSGA-II-A</td>
</tr>
</tbody>
</table>

**TABLE S-V**

<table>
<thead>
<tr>
<th>DC-MOEA</th>
<th>DC-NSGA-II-A</th>
<th>DC-NSGA-II</th>
<th>DC-NSGA-III</th>
<th>DC-TAEA</th>
<th>dCMOEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1079.3</td>
<td>1158.0</td>
<td>1309.2</td>
<td>919.9</td>
<td>16391.7</td>
</tr>
<tr>
<td>2</td>
<td>1668.7</td>
<td>1808.6</td>
<td>2055.4</td>
<td>1470.7</td>
<td>19857.0</td>
</tr>
<tr>
<td>3</td>
<td>2261.4</td>
<td>2276.9</td>
<td>2918.7</td>
<td>2133.1</td>
<td>29274.4</td>
</tr>
<tr>
<td>4</td>
<td>958.9</td>
<td>1142.0</td>
<td>1250.3</td>
<td>829.0</td>
<td>15897.3</td>
</tr>
<tr>
<td>5</td>
<td>1622.6</td>
<td>1795.1</td>
<td>1954.6</td>
<td>1353.8</td>
<td>24041.3</td>
</tr>
<tr>
<td>6</td>
<td>2146.6</td>
<td>2155.2</td>
<td>2845.4</td>
<td>1894.7</td>
<td>30641.0</td>
</tr>
<tr>
<td>7</td>
<td>1035.1</td>
<td>1032.8</td>
<td>1310.9</td>
<td>958.7</td>
<td>15046.7</td>
</tr>
<tr>
<td>8</td>
<td>1790.8</td>
<td>1826.5</td>
<td>2069.9</td>
<td>1441.1</td>
<td>18841.1</td>
</tr>
<tr>
<td>9</td>
<td>2369.8</td>
<td>2052.2</td>
<td>3036.2</td>
<td>2108.4</td>
<td>25982.7</td>
</tr>
<tr>
<td>10</td>
<td>1069.9</td>
<td>1008.0</td>
<td>1208.3</td>
<td>890.5</td>
<td>15981.3</td>
</tr>
<tr>
<td>11</td>
<td>1726.1</td>
<td>1747.7</td>
<td>2031.5</td>
<td>1391.1</td>
<td>24171.7</td>
</tr>
<tr>
<td>12</td>
<td>2388.9</td>
<td>2067.9</td>
<td>2913.0</td>
<td>2037.9</td>
<td>31322.5</td>
</tr>
<tr>
<td>13</td>
<td>986.2</td>
<td>1037.0</td>
<td>1207.4</td>
<td>893.7</td>
<td>17493.0</td>
</tr>
<tr>
<td>14</td>
<td>1569.7</td>
<td>1710.7</td>
<td>1836.7</td>
<td>1361.2</td>
<td>21218.1</td>
</tr>
<tr>
<td>15</td>
<td>2264.7</td>
<td>2080.4</td>
<td>2820.7</td>
<td>1976.7</td>
<td>30785.3</td>
</tr>
<tr>
<td>16</td>
<td>961.2</td>
<td>997.7</td>
<td>1191.4</td>
<td>838.5</td>
<td>17493.0</td>
</tr>
<tr>
<td>17</td>
<td>1506.9</td>
<td>1683.8</td>
<td>1790.2</td>
<td>1292.9</td>
<td>24459.3</td>
</tr>
<tr>
<td>18</td>
<td>2124.6</td>
<td>2237.4</td>
<td>2463.1</td>
<td>1882.8</td>
<td>31751.0</td>
</tr>
<tr>
<td>19</td>
<td>1041.9</td>
<td>1148.5</td>
<td>1231.8</td>
<td>934.0</td>
<td>12833.9</td>
</tr>
<tr>
<td>20</td>
<td>1661.4</td>
<td>1722.1</td>
<td>1902.8</td>
<td>1456.9</td>
<td>18488.9</td>
</tr>
<tr>
<td>21</td>
<td>2232.1</td>
<td>2387.7</td>
<td>3018.7</td>
<td>2473.7</td>
<td>24069.8</td>
</tr>
</tbody>
</table>

**ON THE EIGHT TEST PROBLEMS**

Figures S-5–S-12 show the obtained POFs obtained by the compared algorithms on the eight test problems.
TABLE S-VI
PERFORMANCE METRICS OF DC-MOEA, DC-NSGA-II-A, DC-NSGA-II, DC-NSGA-III, AND DCMOEA WHEN THE TERMINATION CRITERION OF EACH DYNAMIC ENVIRONMENT IS THE ELAPSED CPU TIME LIMIT OF CP=2000MS

<table>
<thead>
<tr>
<th>Instance</th>
<th>DC-MOEA</th>
<th>DC-NSGA-II-A</th>
<th>DC-NSGA-II</th>
<th>DC-NSGA-III</th>
<th>DCMOEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IGD</td>
<td>0.06530(0.00479)‡</td>
<td>0.55941(0.02783)‡</td>
<td>0.47480(0.00217)‡</td>
<td>0.04372(0.00206)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.02100(0.00190)‡</td>
<td>0.28754(0.11983)‡</td>
<td>0.01725(0.00127)‡</td>
<td>0.04920(0.00421)‡</td>
</tr>
<tr>
<td>2</td>
<td>IGD</td>
<td>0.05999(0.00342)‡</td>
<td>0.57744(0.03328)‡</td>
<td>0.04786(0.00267)‡</td>
<td>0.04489(0.00169)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.01803(0.00147)‡</td>
<td>0.27085(0.05832)‡</td>
<td>0.01466(0.00356)‡</td>
<td>0.05629(0.01778)‡</td>
</tr>
<tr>
<td>3</td>
<td>IGD</td>
<td>0.06611(0.00462)‡</td>
<td>0.56390(0.03208)‡</td>
<td>0.04957(0.00243)‡</td>
<td>0.04593(0.00183)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.02381(0.00352)‡</td>
<td>0.22128(0.06707)‡</td>
<td>0.01938(0.00137)‡</td>
<td>0.05496(0.01309)‡</td>
</tr>
<tr>
<td>4</td>
<td>IGD</td>
<td>3.45002(0.00351)‡</td>
<td>1.90126(0.10333)‡</td>
<td>0.03580(0.00844)‡</td>
<td>0.03590(0.00977)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>3.44956(0.00269)‡</td>
<td>1.99611(0.07868)‡</td>
<td>0.03524(0.00786)‡</td>
<td>0.03497(0.00684)‡</td>
</tr>
<tr>
<td>5</td>
<td>IGD</td>
<td>0.06292(0.00208)‡</td>
<td>0.57849(0.03928)‡</td>
<td>0.04551(0.00156)‡</td>
<td>0.04038(0.00158)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.01764(0.00102)‡</td>
<td>0.33137(0.08128)‡</td>
<td>0.03152(0.00112)‡</td>
<td>0.03786(0.01352)‡</td>
</tr>
<tr>
<td>6</td>
<td>IGD</td>
<td>3.43217(0.00083)‡</td>
<td>1.96625(0.06814)‡</td>
<td>0.03688(0.01699)‡</td>
<td>0.03716(0.01992)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.03237(0.00171)‡</td>
<td>0.57517(0.03516)‡</td>
<td>0.02615(0.00427)‡</td>
<td>0.06541(0.00717)‡</td>
</tr>
<tr>
<td>7</td>
<td>IGD</td>
<td>0.07600(0.00220)‡</td>
<td>0.59598(0.03641)‡</td>
<td>0.05788(0.00261)‡</td>
<td>0.05425(0.00108)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.01900(0.00123)‡</td>
<td>0.36330(0.10234)‡</td>
<td>0.01846(0.00154)‡</td>
<td>0.04784(0.01333)‡</td>
</tr>
<tr>
<td>8</td>
<td>IGD</td>
<td>3.46214(0.00329)‡</td>
<td>1.91384(0.05672)‡</td>
<td>0.03542(0.01173)‡</td>
<td>0.03474(0.00337)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.03207(0.00004)‡</td>
<td>0.59786(0.03919)‡</td>
<td>0.05786(0.00266)‡</td>
<td>0.05456(0.00108)‡</td>
</tr>
<tr>
<td>9</td>
<td>IGD</td>
<td>0.07081(0.00279)‡</td>
<td>0.86735(0.03252)‡</td>
<td>0.01532(0.00011)‡</td>
<td>0.03084(0.00341)‡</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.01881(0.00116)‡</td>
<td>0.26879(0.08763)‡</td>
<td>0.01519(0.00108)‡</td>
<td>0.06691(0.01500)‡</td>
</tr>
</tbody>
</table>

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

Fig. S-4. Evolution curves of average IGD values for the third to eighth test problems with n=15 and n=21.
Fig. S-5. Obtained POFs obtained by the compared algorithms on the first test problem.
Fig. S-6. Obtained POFs obtained by the compared algorithms on the second test problem.
Fig. S-7. Obtained POFs obtained by the compared algorithms on the third test problems.
Fig. S-8. Obtained POFs obtained by the compared algorithms on the fourth test problem.
Fig. S-9. Obtained POFs obtained by the compared algorithms on the fifth test problem.
Fig. S-10. Obtained POFs obtained by the compared algorithms on the fifth test problem.
Fig. S-11. Obtained POFs obtained by the compared algorithms on the seventh test problem.
Fig. S-12. Obtained POFs obtained by the compared algorithms on the eighth test problem.
<table>
<thead>
<tr>
<th>Ins</th>
<th>Indicator</th>
<th>DC-NSGA-II</th>
<th>DC-NSGA-III</th>
<th>JCMOE/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HV</td>
<td>2.95610(0.0345)</td>
<td>2.71660(0.0426)</td>
<td>3.06740(0.0443)</td>
</tr>
<tr>
<td></td>
<td>IGD</td>
<td>0.21640(0.0172)</td>
<td>0.28576(0.0155)</td>
<td>0.19700(0.0149)</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.00420(0.0078)</td>
<td>0.00277(0.0056)</td>
<td>0.00004(0.0019)</td>
</tr>
<tr>
<td>2</td>
<td>HV</td>
<td>0.43580(0.0332)</td>
<td>2.63280(0.2948)</td>
<td>3.30640(0.3773)</td>
</tr>
<tr>
<td></td>
<td>IGD</td>
<td>0.18760(0.0123)</td>
<td>0.24690(0.0798)</td>
<td>0.15950(0.0145)</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.06230(0.0147)</td>
<td>0.24800(0.0405)</td>
<td>0.08310(0.0037)</td>
</tr>
<tr>
<td>3</td>
<td>HV</td>
<td>0.22490(0.0343)</td>
<td>0.28730(0.1364)</td>
<td>0.19550(0.0224)</td>
</tr>
<tr>
<td></td>
<td>IGD</td>
<td>0.08730(0.0467)</td>
<td>0.24890(0.0282)</td>
<td>0.04710(0.0825)</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.04170(0.0292)</td>
<td>0.28250(0.1599)</td>
<td>0.13460(0.1398)</td>
</tr>
<tr>
<td>4</td>
<td>HV</td>
<td>0.18360(0.0073)</td>
<td>0.24770(0.0949)</td>
<td>0.16630(0.0388)</td>
</tr>
<tr>
<td></td>
<td>IGD</td>
<td>0.06130(0.0163)</td>
<td>0.22420(0.0474)</td>
<td>0.03720(0.0285)</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.26050(0.0461)</td>
<td>0.27230(0.0366)</td>
<td>0.03020(0.0146)</td>
</tr>
<tr>
<td>5</td>
<td>HV</td>
<td>0.23490(0.0134)</td>
<td>0.29480(0.0124)</td>
<td>0.19670(0.0171)</td>
</tr>
<tr>
<td></td>
<td>IGD</td>
<td>0.08960(0.0174)</td>
<td>0.26740(0.0513)</td>
<td>0.04820(0.0282)</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.03250(0.0391)</td>
<td>0.28200(0.0301)</td>
<td>0.17000(0.0427)</td>
</tr>
<tr>
<td>6</td>
<td>HV</td>
<td>0.19370(0.0106)</td>
<td>0.25300(0.0892)</td>
<td>0.15870(0.0768)</td>
</tr>
<tr>
<td></td>
<td>IGD</td>
<td>0.04230(0.0261)</td>
<td>0.24500(0.0094)</td>
<td>0.14560(0.0615)</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.09210(0.0308)</td>
<td>0.27190(0.0490)</td>
<td>0.10720(0.0590)</td>
</tr>
<tr>
<td>7</td>
<td>HV</td>
<td>0.23210(0.1205)</td>
<td>0.29670(0.1335)</td>
<td>0.20460(0.0238)</td>
</tr>
<tr>
<td></td>
<td>IGD</td>
<td>0.08350(0.0129)</td>
<td>0.23760(0.0403)</td>
<td>0.09450(0.0621)</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.02140(0.0148)</td>
<td>0.28170(0.0327)</td>
<td>0.13080(0.0397)</td>
</tr>
<tr>
<td>8</td>
<td>HV</td>
<td>0.33910(0.1098)</td>
<td>0.25190(0.0866)</td>
<td>0.18680(0.1048)</td>
</tr>
<tr>
<td></td>
<td>IGD</td>
<td>0.07630(0.0099)</td>
<td>0.19310(0.0086)</td>
<td>0.06840(0.0218)</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>0.02360(0.0140)</td>
<td>0.23990(0.0474)</td>
<td>0.04030(0.0372)</td>
</tr>
</tbody>
</table>

S-X. Three Performance Metrics of Algorithms on the Test Problems with 15 and 20 Decision Variables

Tables S-VII–S-IX give the values and standard deviations of the three performance metrics (i.e., HV, IGD, and MS) for these algorithms (i.e., DC-NSGA-II, DC-NSGA-III, and JCMOE/A) on the test problems with 15 decision variables. The results of these algorithms on the test problems with 20 decision variables are given in Tables S-X–S-XII.


<table>
<thead>
<tr>
<th>Table S XI</th>
<th></th>
<th>Table S XII</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INS</strong></td>
<td><strong>DC-NSGA-II</strong></td>
<td><strong>DC-NSGA-III</strong></td>
</tr>
<tr>
<td><strong>HV</strong></td>
<td>2.99175(0.03964)</td>
<td>2.70604(0.05468)</td>
</tr>
<tr>
<td><strong>IGD</strong></td>
<td>0.21273(0.0148)</td>
<td>0.20990(0.01775)</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>0.04755(0.00557)</td>
<td>0.30048(0.00827)</td>
</tr>
<tr>
<td><strong>HV</strong></td>
<td>3.10505(0.02876)</td>
<td>2.86166(0.02986)</td>
</tr>
<tr>
<td><strong>IGD</strong></td>
<td>0.16964(0.00899)</td>
<td>0.24754(0.00969)</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>0.03864(0.00844)</td>
<td>0.30651(0.07435)</td>
</tr>
<tr>
<td><strong>HV</strong></td>
<td>2.98844(0.03836)</td>
<td>2.72530(0.03521)</td>
</tr>
<tr>
<td><strong>IGD</strong></td>
<td>0.21324(0.01382)</td>
<td>0.29033(0.01852)</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>0.05736(0.01671)</td>
<td>0.32764(0.08137)</td>
</tr>
<tr>
<td><strong>HV</strong></td>
<td>3.01955(0.02422)</td>
<td>2.85412(0.03503)</td>
</tr>
<tr>
<td><strong>IGD</strong></td>
<td>0.16652(0.00845)</td>
<td>0.24521(0.01194)</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>0.04025(0.00953)</td>
<td>0.31903(0.07438)</td>
</tr>
<tr>
<td><strong>HV</strong></td>
<td>2.95032(0.01051)</td>
<td>2.72222(0.03509)</td>
</tr>
<tr>
<td><strong>IGD</strong></td>
<td>0.22552(0.01547)</td>
<td>0.30830(0.01672)</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>0.05314(0.01096)</td>
<td>0.34773(0.06148)</td>
</tr>
<tr>
<td><strong>HV</strong></td>
<td>0.17503(0.01153)</td>
<td>0.25629(0.01275)</td>
</tr>
<tr>
<td><strong>IGD</strong></td>
<td>0.04318(0.00542)</td>
<td>0.30720(0.08988)</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>0.05885(0.01186)</td>
<td>0.34773(0.06148)</td>
</tr>
<tr>
<td><strong>HV</strong></td>
<td>2.9703(0.04380)</td>
<td>2.70560(0.04485)</td>
</tr>
<tr>
<td><strong>IGD</strong></td>
<td>0.22356(0.01541)</td>
<td>0.31223(0.01763)</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>0.05207(0.00824)</td>
<td>0.34496(0.08266)</td>
</tr>
<tr>
<td><strong>HV</strong></td>
<td>3.07442(0.04158)</td>
<td>2.84352(0.03452)</td>
</tr>
<tr>
<td><strong>IGD</strong></td>
<td>0.17324(0.01034)</td>
<td>0.25339(0.01083)</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>0.04032(0.00354)</td>
<td>0.32526(0.07144)</td>
</tr>
</tbody>
</table>