Decoupling Constraint from Two Direction in Evolutionary Constrained Multi-objective Optimization

Ruiqing Sun, Dawei Feng, Xing Zhou, Lianghao Li, Sheng Qi, Bo Ding, Yijie Wang, Rui Wang Senior Member, IEEE, Huaimin Wang

Abstract—Real-world Constrained Multi-objective Optimization Problems (CMOPs) often contain multiple constraints, and understanding and utilizing the coupling between these constraints is crucial for solving CMOPs. However, existing Constrained Multi-objective Evolutionary Algorithms (CMOEAs) typically ignore these couplings and treat all constraints as a single aggregate, which lacks interpretability regarding the specific geometric roles of constraints. To address this limitation, we first analyze how different constraints interact and show that the final Constrained Pareto Front (CPF) depends not only on the Pareto fronts of individual constraints but also on the boundaries of infeasible regions. This insight implies that CMOPs with different coupling types must be solved from different search directions. Accordingly, we propose a novel algorithm named Decoupling Constraint from Two Directions (DCF2D). This method periodically detects constraint couplings and spawns an auxiliary population for each relevant constraint with an appropriate search direction. Extensive experiments on seven challenging CMOP benchmark suites and on a collection of real-world CMOPs demonstrate that DCF2D outperforms five state-of-the-art CMOEAs, including existing decoupling-based methods. Code available at: https://github.com/KFC-Grandpa/DCF2D-Decoupling-Constraint-from-Two-Direction

Index Terms—Constraint handling, Coevolutionary algorithm, Constraint decoupling.

I. Introduction

Many real-world engineering and scientific applications require optimizing several conflicting objectives under multiple constraints [1], [2], [3]. This class of problems is referred to as Constrained Multi-objective Optimization Problems (CMOPs). Solving CMOPs requires obtaining a set of decision vectors that simultaneously optimize multiple conflicting objectives while satisfying constraints.

Ruiqing Sun, Dawei Feng, Bo Ding, Yijie Wang, and Huaimin Wang are with the College of Computer Science and Technology, National University of Defense Technology, Changsha 410000, P.R. China (e-mail: sunny0331@foxmail.com; davyfeng.c@qq.com; dingbo@nudt.edu.cn; wangyijie@nudt.edu.cn, hmwang@nudt.edu.cn).

Xing Zhou is with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha, 410073, P.R. China (e-mail: zhouxing@nudt.edu.cn).

Lianghao Li is with the State Key Laboratory of Complex & Critical Software Environment, College of Information and Communication, National University of Defense Technology, Wuhan 430019, P.R. China (e-mail: lianghao93@nudt.edu.cn;).

Sheng Qi and Rui Wang are with the College of System Engineering, National University of Defense Technology, Changsha 410000, P.R. China (e-mail: qisheng@nudt.edu.cn; ruiwangnudt@gmail.com)

Without loss of generality, a CMOP can be defined as follows [4]:

$$
\begin{cases}
\min\ F(X) = (f_1(X), f_2(X), \ldots, f_M(X)), \\
subject\ to\ X \in \Omega, \\
g_i(X) \leq 0, \quad i = 1, \ldots, p \\
h_j(X) = 0, \quad j = 1, \ldots, q \\
n_{con} = p + q
\end{cases}
\tag{1}
$$

where $X = (x_1, x_2, \ldots, x_D)$ is a D-dimensional decision variable vector from the decision space Ω; $F(X)$ is an objective function vector composed of M conflicting objective functions; $g_i(X)$ are p inequality constraints and $h_j(X)$ are q equality constraints. Among them, n_{con} is the number of all constraints, equal to $p + q$.

In CMOPs, there are not only feasible solutions but also infeasible solutions in which the satisfaction of each constraint can be measured by $c(X)$ [5]:

$$
c_k(X) =
\begin{cases}
\max(0, g_k(X)), & k = 1, \ldots, p \\
\max(0, |h_{k-p}(X) - \delta|), & k = p + 1, \ldots, n_{con}
\end{cases}
\tag{2}
$$

where δ is a small enough relaxation factor to relax equality constraints. By summing up the violation degrees of each constraint, denoted as $CV(X)$, as shown in Equation (3), we can determine the overall constraint violation of a solution. If the value of $CV(X)$ is zero, the solution is considered feasible; otherwise, it is deemed infeasible.

$$
CV(X) = \sum_{k=1}^{n_{con}} c_k(X),
\tag{3}
$$

CMOPs have been proven to be NP-hard problems [6]. The ultimate goal of solving a CMOP is to obtain a set of feasible solutions

$$
S = \{ X^1, X^2, \ldots, X^k \}
\tag{4}
$$

such that, for every $X \in S$, there does not exist any other feasible solution Y satisfying the following Pareto-dominance relation $Y \prec_F X$ defined as

$$
\begin{cases}
f_i(X) \geq f_i(Y), & \forall i = 1, 2, \ldots, M, \\
f_j(X) > f_j(Y), & \exists j \in \{1, 2, \ldots, M\}, \\
CV(Y) = 0, \quad CV(X) = 0.
\end{cases}
\tag{5}
$$

In other words, Y is no worse than X in every objective and strictly better in at least one objective, while both Y and X satisfy all constraints. The set S of such non-dominated feasible solutions is called the Constrained Pareto Front (CPF):

$$\mathcal{F} = \{\, X \in \Omega \mid CV(X) = 0 \}, \tag{6}$$

$$\text{CPF} = \{\, X \in \mathcal{F} \mid \neg \exists Y \in \mathcal{F} : \ Y \prec_F X \}. \tag{7}$$

When the feasibility requirement is dropped—i.e., we consider Pareto-dominance over the entire decision space Ω regardless of constraint violations—the corresponding set of non-dominated points is called the Unconstrained Pareto Front (UPF):

$$\text{UPF} \ = \ \{\, X \in \Omega \mid \neg \exists Y \in \Omega : Y \prec X \}, \tag{8}$$

where "$Y \prec X$" denotes the standard Pareto-dominance relation (strictly better in at least one objective and no worse in all others) without considering constraint satisfaction.

Approximating the CPF requires algorithms to consider feasibility, convergence, and diversity simultaneously. Multi-objective Evolutionary Algorithms (MOEAs) aim to optimize solutions to satisfy both convergence and diversity, while Constraint Handling Techniques (CHTs) enable MOEAs to incorporate feasibility during optimization. The combination of MOEAs and CHTs forms Constrained Multi-objective Evolutionary Algorithms (CMOEAs). In extensive practice, CMOEAs have proven to be effective and efficient methods for approximating the CPF [7], [8], [9].

Over the past twenty years, many CHTs have been proposed, most of which are based on constraint violation (Equation (3)). For example, the ϵ-constrained method relaxes constraints by considering solutions with $CV(X)$ less than ϵ as pseudo-feasible solutions; multi-objective methods optimize $CV(X)$ values as additional objectives; penalty-based methods penalize solutions based on the magnitude of $CV(X)$; and the Constraint Dominance Principle (CDP) prioritizes feasible solutions. These CHTs naturally overlook the coupling relationships between constraints, treating all constraints as a single aggregate. When a CMOP has multiple constraints, the coupling between them may form a complex environment, posing distinct challenges regarding feasibility, convergence, and diversity. Therefore, treating constraints merely as a scalar $CV(X)$ naturally overlooks these coupling relationships, potentially hindering the solving process.

Moreover, this aggregation approach renders the constraint handling process a "black box." It obscures the specific geometric role of each constraint—whether it blocks convergence to the Pareto front or merely reduces the feasible space volume. In contrast, analyzing constraint coupling offers a "white-box" perspective with higher interpretability. By explicitly identifying which constraints are coupled and how they interact to form the final CPF, we can not only solve the problem more efficiently but also provide decision-makers with valuable insights into the problem's structure.

Recently, researchers have recognized this issue and proposed constraint decoupling methods, such as MSCMO [10], C3M [11], MCCMO [12], and MTOTC [13]. These methods attempt to analyze the roles of specific constraints or decompose them into sub-problems to improve convergence. While these approaches have demonstrated superiority over traditional aggregation methods on complex CMOPs, existing decoupling methods still possess certain limitations:

1) The CPF of a CMOP with multiple constraints may not be related to all constraints. However, except for C3M, all constraints are usually considered for decoupling.
2) The decoupling direction of these methods typically follows the direction of evolution, i.e., from the dominated region to the non-dominated region (e.g., from the upper right corner to the bottom left corner in Fig. 1(a)), searching for the CPF of single constraints (SCPF).

However, we observed that in some CMOPs, the final CPF is not always formed by the SCPF of single constraints (as shown in Fig. 1(d)). Therefore, constraint decoupling should not be carried out solely from one direction.

Here, for any two solutions $X, Y \in \Omega$, we say the direction from X to Y is a positive direction if $Y \prec X$. Conversely, the vector from X to Y is a negative direction if $X \prec Y$. In addition, we define the CPF of constraint i as SCPF_i:

$$F_i \ = \ \{\, X \in \Omega \mid c_i(X) = 0 \}, \tag{9}$$

$$\text{SCPF}_i \ = \ \{\, X \in F_i \mid \neg \exists Y \in F_i : \ Y \prec_{F_i} X \}. \tag{10}$$

For the part of the CPF that is not related to any SCPF, we define it as ICPF:

$$\text{ICPF} \ = \ \text{CPF} \ \setminus \ \bigcup_{i=1}^{n_{con}} \text{SCPF}_i \tag{11}$$

Intuitively, the ICPF can be described as the portion of the true Constrained Pareto Front that is unreachable by solely optimizing within the feasible region of any single constraint. Finally, for the infeasible solutions in the i-th constraint, the solution set consisting of solutions closest to the ICPF is defined as RCPF_i:

$$G_i \ = \ \{\, X \in \Omega \mid c_i(X) > 0 \}, \tag{12}$$

$$\begin{aligned} \text{RCPF}_i \ = \ \{\, X \in G_i \ \mid \ &(\exists Y \in \text{ICPF} : \ Y \prec X) \\ &\wedge (\neg \exists Z \in G_i : \ Z \prec X) \}. \end{aligned} \tag{13}$$

The RCPF serves as a 'stepping stone' for discovering the ICPF, representing the set of infeasible solutions that are nearest to it. Note that when the ICPF does not exist, the RCPF is always an empty set.

As shown in Fig. 1, when the CMOP contains multiple constraints, there are four situations:

1) The CPF of the CMOP is related to the two SCPFs. In this case, the SCPF of the relevant constraints is helpful to the final CPF. As shown in Fig. 1(a), the final CPF can be obtained while searching the CPF
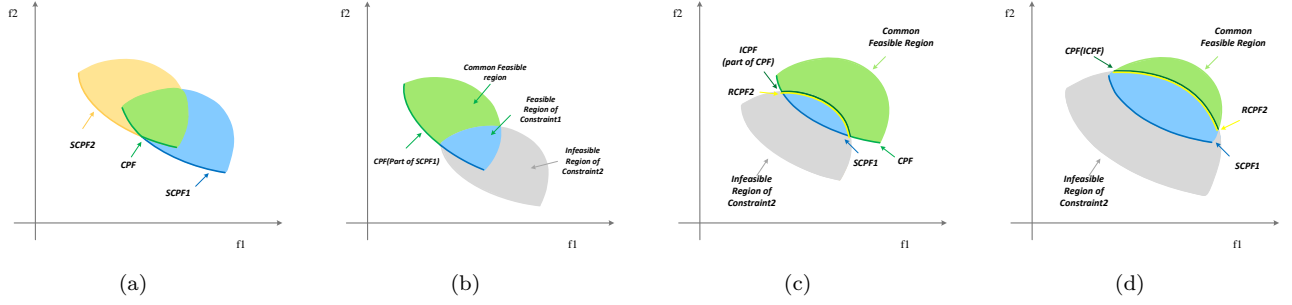
Fig. 1. Four types of coupling constraints. (a) The final CPF is composed of parts of two SCPFs. (b) The final CPF is composed of only a single SCPF, and the infeasible region of another constraint makes this SCPF partly infeasible. (c) The final CPF is partially composed of a SCPF, and the other part of the CPF (ICPF) is affected by the infeasible region of another constraint and is closely connected to the boundary of the infeasible region (we define it as RCPF). (d) Affected by the infeasible region of one of the constraints, the entire final CPF is ICPF that is only related to the RCPF.

of Constraint 1 and Constraint 2 from the positive direction.

2) The CPF of the CMOP is related only to one SCPF. However, part of the SCPF of this constraint is blocked by the infeasible region of other constraints. The CPF of the relevant constraints is helpful to the final CPF. As shown in Fig. 1(b), the final CPF can be obtained while searching for SCPF1 from the positive direction, and constraint 2 is less helpful.

3) The CPF of the CMOP is related to the SCPF and RCPF of different constraints. In this case, constraints need to be solved from both directions. For a constraint whose CPF is related to its SCPF, searching from the positive direction to find its SCPF is more helpful, while for a constraint whose CPF is related to its RCPF, searching for its ICPF from the negative direction is more helpful. As shown in Fig. 1(c), searching for SCPF1 from the positive direction is helpful for the final CPF, while searching for RCPF2 from the negative direction is helpful for the ICPF.

4) The entire CPF is the ICPF, and related to the RCPF. In this case, searching for the SCPF is less helpful for the final CPF, while searching for the RCPF is more helpful for the final CPF. As shown in Fig. 1(d), searching for SCPF1 from the positive direction offers no help for the final CPF, while searching for RCPF2 from the negative direction is more helpful.

Motivated by the observations above, we have designed a novel decoupled CMOEA called DCF2D (i.e., Decoupling Constraint from Two Directions). Firstly, to ensure convergence, an auxiliary population that completely disregards feasibility will explore the objective space, striving to find the UPF of the CMOP as much as possible. This is very helpful because the UPF may (completely or partially) be a part of the final CPF. If not, this auxiliary population turns to exploring the infeasible region in front of the main population. This exploration is only based on diversity, thus ensuring both convergence and diversity. For the main population, a newly designed strategy periodically detects the relation-

ship between constraints related to the main population and determines how many auxiliary populations to spawn and the search direction of those populations based on the relationship between constraints. The experimental results show that decoupling from two directions improves the final solution's quality more than decoupling from one direction.

The contributions of this paper are summarized as follows:

1) We systematically analyze how different constraints in the same CMOP are coupled to form the final CPF. We demonstrate that the final CPF may not be related to all SCPFs, leading to the proposal of the ICPF concept. Furthermore, to solve for the ICPF, we introduce the concept of RCPF, which represents the set of infeasible solutions closest to the ICPF in the objective space.

2) We propose a dynamic detection method for constraint coupling relationships. This method identifies which constraints' SCPF or RCPF contribute to the final CPF during the evolutionary process. By focusing only on relevant constraints, the algorithm avoids processing unnecessary constraints, thereby reducing computational overhead.

3) Based on these insights, we propose a novel CMOEA named DCF2D. The algorithm starts by exploring the unconstrained problem and gradually transitions to the constrained problem. Throughout this process, it utilizes constraint coupling information to spawn auxiliary populations with specific search directions (positive or negative) for helpful constraints. Finally, it guides the population back to the original CMOP to obtain the final feasible solutions. Extensive experiments demonstrate that DCF2D achieves superior performance in solving complex CMOPs with multiple constraints.

The remainder of this paper is organized as follows. Section II briefly reviews related work. Section III presents the proposed algorithm in detail. Section IV describes the experimental setup and analyzes the results. Finally, Section V concludes the paper and discusses limitations and future prospects.

TABLE I
List of abbreviations and symbols used throughout the paper

Abbreviation	Meaning
n_{con}	Number of constraints
CMOP	Constrained Multi-objective Optimization Problem
MOEA	Multi-objective Evolutionary Algorithm
CHT	Constraint Handling Technique
CMOEA	Constrained MOEA
CPF	Constrained Pareto Front
UPF	Unconstrained Pareto Front
CV	Constraint Violation
CDP	Constraint Dominance Principle
SCPF	Single-Constraint Pareto Front
ICPF	Part of CPF excluding the union of all SCPFs
RCPF	Nearest infeasible Boundary to ICPF
Positive Direction	Search direction from solution X to Y where $Y \prec X$
Negative Direction	Search direction from solution X to Y where $X \prec Y$

II. Related Works

Based on the constraint handling mechanism and the evolutionary framework, existing CMOEAs can be broadly categorized into five types: Traditional Constraint Handling Techniques (CHTs), Co-evolutionary (or Multi-population) methods, Multi-stage methods, Machine Learning-assisted methods, and Constraint Decoupling methods. To provide a clearer comparison, Table II summarizes the key features of representative state-of-the-art algorithms and the proposed DCF2D.

A. Traditional Constraint Handling Techniques

Most early CMOEAs employ a single population and handle constraints by aggregating them into a single metric, typically the overall Constraint Violation (CV). The Constraint Dominance Principle (CDP) [15] is the most widely used method. The ϵ-constrained method relaxes the feasibility requirement dynamically. Penalty function methods [16], [17] add a penalty term based on $CV(X)$ to the objective function. Other approaches, such as Stochastic Ranking [18] and Multi-Objective Based frameworks [19], balance objective optimization and constraint satisfaction. MOEAD-CPBI [20] integrates the normalized overall constraint violation into decomposition-based algorithms. While effective for simple constraints, these methods treat all constraints as a whole, thus ignoring the complex coupling relationships between individual constraints.

B. Co-evolutionary and Multi-population Methods

To overcome the limitations of single-population search, researchers have introduced co-evolutionary frameworks that utilize auxiliary populations (or archives) to assist the main population. CCMO [4] maintains a main population and an auxiliary population (solving the MOP) to bypass infeasible barriers. Following this, various dual-population or multi-population mechanisms have been proposed to enhance interaction and diversity. For instance, EMCMO [21], CMOSMA [22], and C-TAEA [23] improve information transfer strategies. ACREA [24] proposes a novel co-evolutionary framework with two archives. Specifically, it maintains a leading archive that employs an Adaptive Constraint Relaxation (ACR) strategy. By dynamically determining whether to consider constraints based on the relationship between the two archives, the leading archive guides the primary archive to traverse large infeasible regions and avoid local optima. APSEA [25] introduces an adaptive population sizing strategy. It dynamically shrinks the auxiliary population based on the evolutionary status and employs a multi-stage constraint handling mechanism to maximize search efficiency. These methods leverage the diversity of auxiliary populations but typically still treat constraints as a single aggregate during the environmental selection process. Other methods, such as BiCo [26], cDPEA [27], CCMODE [28], CMOEA-TCP [29], DBC-CMOEA [30], Dp-ACS [31], TPEA [32], EMCMMS [33], and CMOEA-CD [34], introduce different auxiliary populations or archiving strategies to balance convergence and diversity.

C. Multi-stage Methods

Multi-stage methods divide the evolutionary process into distinct phases with different optimization goals. Typically, the first stage focuses on exploring the UPF or finding feasible regions, while the second stage focuses on converging to the CPF. Classic examples include ToP [35], MOEA/D-DAE [36], TSTI [37], CMOEA_MS [38], TSCSO [39], and CMOES [40]. Some methods introduce transitional stages or more complex phase switching, such as CMOEMT [41], CAEAD [42], MSCEA [43], MTCMO [44], and TriP [45]. MOEA/D-LCDP [46] combines the local constraint dominance principle to balance constraint satisfaction. While effective in navigating infeasible regions, these methods often rely on handcrafted switching strategies and may struggle when the UPF and CPF are disjoint or when constraints are highly coupled.

D. Machine Learning-assisted Methods

With the advancement of artificial intelligence, an increasing number of methods are incorporating machine learning techniques to enhance the adaptability

TABLE II
Comparison of features between DCF2D and representative state-of-the-art CMOEAs

Algorithm	Year	Category	Constraint Handling	Decoupling Strategy	Search Direction	Key Mechanism
CCMO [4]	2019	Co-evolutionary	Aggregate (CDP)	No	Uni-directional	Population Sharing
MSCMO [10]	2021	Decoupling	Progressive	Yes (Sequential)	Uni-directional	Infeasibility Rate
C3M [11]	2022	Decoupling	Classification	Yes (Rank)	Uni-directional	Constraint Classification
MCCMO [12]	2023	Decoupling	Decomposition	Yes (Combine)	Uni-directional	Knowledge Transfer
MTOTC [13]	2024	Decoupling	Decomposition	Yes (One-off)	Uni-directional	Task Cloning
FCDS [14]	2025	Decoupling	Fuzzy Dominance	Yes (Level-based)	Uni-directional	Adaptive Difficulty
DCF2D (Proposed)	2026	Decoupling	Bi-directional (SCPF & RCPF)	Yes (Dynamic)	Bi-directional	Coupling Detection & Auxiliary Populations

of CMOEAs. RLSD [47] (2025) utilizes Reinforcement Learning (RL) to dynamically determine the evolutionary stage based on the population state. CMOQLMT [48] uses Q-learning to select the most suitable auxiliary optimization strategy during evolution. Other methods like DBEMTO [49], DD-CMOEA [50], DVCEA [51], and GC-PPS [52] employ adaptive policies, classification, or clustering techniques to assist optimization. Although these methods improve robustness through learning, they often function as "black boxes" and entail additional training costs. Moreover, they typically learn implicit patterns rather than explicitly analyzing the geometric coupling structure of constraints.

E. Constraint Decoupling Methods

Constraint decoupling has emerged as a promising direction to handle complex constraints by analyzing the specific role of individual constraints explicitly. Unlike black-box methods, constraint decoupling offers higher interpretability, allowing researchers to understand which constraints specifically hinder convergence or form the Pareto front. MSCMO [10] determines constraint priority based on infeasibility rates. C3M [11] analyzes the relationship between the CPFs of different constraints. Similarly, FCDS [14] addresses the limitations of aggregating constraint violations by introducing a fuzzy dominance strategy. It decouples the constraint assessment into discrete satisfaction levels , effectively transforming the complex CMOP into a sequence of sub-problems with increasing difficulty to bypass local optima. MCCMO [12] employs a multi-population framework where each sub-population optimizes a decoupled sub-problem. MTOTC [13] and IMTCMO [53] generate auxiliary populations that ignore specific constraints to assist the main population.

However, existing decomposition-based decoupling methods (like MTOTC and IMTCMO) primarily search from the feasible direction (approximating the SCPF). Although methods like FCDS utilize infeasible solutions in early stages, they rely on global relaxation rather than explicit boundary targeting. As analyzed in Section I, for constraints where the CPF is formed by the infeasible boundary (RCPF), a targeted search is required. DCF2D addresses this by decoupling constraints from both positive (SCPF) and negative (RCPF) directions based on explicit coupling detection.

III. PROPOSED ALGORITHM

A. Framework of DCF2D

The proposed DCF2D is a multi-population co-evolutionary framework designed to solve CMOPs by explicitly decoupling constraints. It operates in three adaptive stages: Global Exploration, Co-evolution with Decoupling, and Final Convergence. The complete procedure is outlined in Algorithm 1, and the key variables used in the algorithm are defined as follows:

- MP: The Main Population (N individuals) responsible for finding the feasible CPF.

- AP_i: The i-th Auxiliary Population. AP_0 explores the UPF (N individuals), while $AP_{1...n_{con}}$ handles specific constraints ($N/4$ individuals).
- LI: The Infeasible Archive, storing boundary solutions that dominate the MP in objective space.
- D_i: The search direction for AP_i. $True$ denotes Positive (searching for SCPF), and $False$ denotes Negative (searching for RCPF).
- Act_i: A boolean flag indicating whether AP_i is active.
- $WindowSize$: The size of the sliding window used for convergence detection (set to 5).

1. Initialization and Reproduction (Lines 1-3): Initially, MP and AP_0 are generated randomly. To balance computational efficiency and search diversity, we employ a Dynamic Resource Allocation Strategy. In each generation, the total number of offspring is fixed at N. The MP is assigned 50% of the budget ($N/2$) to ensure consistent convergence pressure. The remaining 50% is distributed equally among all active Auxiliary Populations (AP_i). To guarantee the effectiveness of the Differential Evolution (DE) operators, a minimum quota (e.g., 5 offspring) is enforced for each active AP_i.

The adoption of the hybrid DE operators extends beyond ensuring a fair comparison with state-of-the-art methods like IMTCMO; it is fundamentally driven by their suitability for constraint decoupling populations. Specifically, DE/rand/1 provides robust global exploration, which is critical for identifying disjoint feasible regions formed by complex constraints. Meanwhile, DE/current-to-pbest/1 utilizes difference vectors pointing toward superior individuals, generating strong directional pressure. This mechanism effectively drives the auxiliary population to rapidly approximate and tightly 'hug' the specific boundaries (SCPF or RCPF) of the decoupled constraints, a geometric capability that is essential for providing accurate guidance to the main population.

2. Stage 1: Global Exploration via Trajectory Analysis (Lines 10, 12-16): In this stage, AP_0 ignores all constraints to approximate the UPF. To robustly detect the convergence of AP_0 and avoid premature transition due to evolutionary noise, we introduce a Sliding Window Trajectory Analysis. Let $\mathcal{C}_t$ be the centroid of AP_0 in the objective space at generation t. The relative movement M_t is defined as:

$$M_t = \frac{\|\mathcal{C}_t - \mathcal{C}_{t-1}\|}{\|\mathcal{C}_t\| + \epsilon}, \tag{14}$$

where ϵ is a small constant to avoid division by zero. We maintain a sliding window of size $W = 5$ to record the history of movements. The transition to Stage 2 is triggered only when the average movement over the window falls below a threshold δ:

$$\frac{1}{W} \sum_{k=0}^{W-1} M_{t-k} < \delta \quad (\text{e.g., } \delta = 10^{-4}). \tag{15}$$

Upon transition, AP_0 is deactivated to save resources, and an initial scan of LI activates the constraint-specific populations.

Algorithm 1: Framework of DCF2D

Input: N (population size), α (detection interval), β (stage 3 threshold)
Output: MP (Final solutions)

1 Initialize: $n_{con} \leftarrow$ constraint count; $MP \leftarrow$ Init(N); $AP_{0...n_{con}} \leftarrow \emptyset$;
2 State: $Stage \leftarrow 1$; $Act_0 \leftarrow True$; $Act_{1...n_{con}} \leftarrow False$; $D_{1...n_{con}} \leftarrow Positive$;
3 $AP_0 \leftarrow MP$; $LI \leftarrow \emptyset$;
4 **while** termination criterion not fulfilled **do**
5   Reproduction (Dynamic Resource Allocation);
6   Allocate offspring budget: 50% to MP, 50% shared among active AP_i;
7   Generate offspring set $\mathcal{O}$ using DE operators;
8   Update Populations;
9   Update MP using $\mathcal{O}$ based on CDP;
10   Update AP_0 using $\mathcal{O}$ based on raw objective values;
11   Update active $AP_{1...n_{con}}$ using $\mathcal{O}$ in direction D_i (Alg. 3);
12   Update LI using $\mathcal{O}$ (Alg. 2);
     // Event-Driven Resurrection Mechanism
13   **if** LI is updated and Stage ≥ 2 **then**
14      $infCon \leftarrow$ Detect violated constraints in new LI solutions;
15   **foreach** $j \in infCon$ **do**
16   **if** AP_j is inactive **then**
17          $Act_j \leftarrow True$; $D_j \leftarrow Positive$; Set Protection for AP_j;

18   **switch** $Stage$ **do**
19   **case** 1 (Exploration) **do**
20   Calculate Centroid Movement ΔM_t of AP_0 (Eq. 15);
21   **if** Average Movement over Window $< 10^{-4}$ **then**
22          $Stage \leftarrow 2$; $Act_0 \leftarrow False$; // Stop Global Exploration
23          $infCon \leftarrow$ Detect violated constraints in LI;
24   Activate AP_i for all $i \in infCon$;

25   **case** 2 (Co-evolution with Decoupling) **do**
26   **foreach** active AP_i ($i \in \{1...n_{con}\}$) **do**
         // Dynamic Direction Adjustment
27   **if** D_i is $Positive$ and no feasible solution in AP_i **then**
28            $D_i \leftarrow Negative$; // Switch to RCPF search
29   **else if** D_i is $Negative$ and AP_i dominated by MP and not Protected **then**
30            $Act_i \leftarrow False$; // Deactivate

31   **case** 3 (Convergence) **do**
32   **if** $FE > \beta \times MaxFE$ **then** Deactivate all AP_i; $Stage \leftarrow 3$;

33 **return** MP;

---

3. Stage 2: Event-Driven Co-evolution (Lines 9, 11, 17-23): This is the core stage where constraints are decoupled. DCF2D employs an Event-Driven Mechanism:

- Resurrection (Activation): Whenever Algorithm 2 updates LI with new boundary solutions, we check which constraints they violate. If a constraint j is

violated but AP_j is inactive, AP_j is immediately resurrected with a temporary protection period. This ensures the algorithm responds instantly to new barriers encountered by the MP.

- Dynamic Adjustment (Deactivation): Active populations are monitored. If an AP_i searching in the Positive direction finds no feasible solutions, it switches to Negative to target the RCPF. Conversely, if a Negative-direction population becomes dominated by the MP (indicating the barrier is bypassed) and its protection period has expired, it is deactivated.

4. Stage 3: Final Convergence (Line 30-31): When the evaluation count exceeds $\beta \times MaxFE$, all auxiliary populations are permanently stopped. All resources are concentrated on the MP to refine the precision of the final feasible solutions.

B. Key functions in DCF2D

Algorithm 2: Update Infeasible Archive (LI)

Input: LI (Current Archive), $\mathcal{O}$ (Offspring), N (Archive Size), MP (Main Population)
Output: LI (Updated Archive)

1 $S \leftarrow LI \cup \mathcal{O}$;
   // Filter 1: Objective-Dominance Screening
   // Retain s if it is better than MP in objectives but strictly dominated in constraints (implied by infeasibility)
2 $S \leftarrow \{s \in S \mid (\exists x \in MP : s \prec_{obj} x) \land (\nexists y \in MP : y \prec_{obj} s)\}$;
3 $S_{infeas} \leftarrow$ Select all infeasible solutions in S;
4 $S_{feas} \leftarrow$ Select all feasible solutions in S;
   // Filter 2: Boundary-Focused Selection
5 **if** $|S_{infeas}| > N$ **then**
     // Prioritize solutions closer to the CPF boundary
6   Calculate fitness of S_{infeas} based on negative objective values ($-F(x)$);
7    $LI \leftarrow$ Select top N solutions from S_{infeas} based on the calculated fitness;
8 **else**
9    $LI \leftarrow S_{infeas}$;
10    $k \leftarrow N - |S_{infeas}|$;
     // Fill remaining slots with best feasible solutions
11   Calculate fitness of S_{feas} based on normal objective values ($F(x)$);
12    $LI \leftarrow LI\cup$ Select top k solutions from S_{feas};
13 **return** LI;

---

1) Update LI: Algorithm 2 details the update procedure for the Infeasible Archive (LI). Crucially, LI acts as a "radar" for the Resurrection Mechanism in the main framework. By maintaining infeasible solutions that dominate the MP in the objective space, it identifies the specific constraints forming the boundaries (SCPF or RCPF) that currently block convergence. The update process involves two rigorous filtering steps:

Step 1: Objective-Dominance Screening. First, the existing archive is merged with the offspring. We strictly retain only those solutions that dominate at least one solution in the current MP in terms of objective values
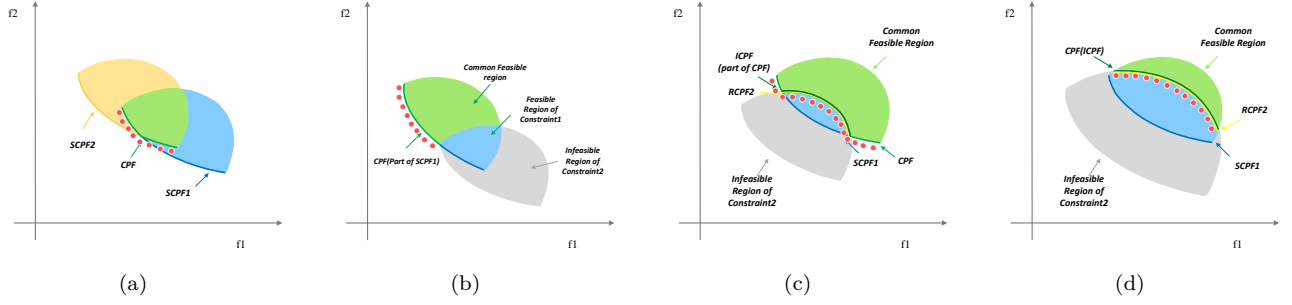
Fig. 2. The distribution of solutions in the Infeasible Archive (LI, marked as red circles) across four situations. LI contains non-dominated infeasible solutions found during evolution, which are used to detect constraints that block the convergence of the CPF. Identifying the infeasible boundaries (RCPF) or SCPFs of these blocking constraints is crucial for guiding the search direction in DCF2D.

($s \prec_{obj} x$) but are not dominated by any solution in the MP ($y \not\prec_{obj} s$). This logic filters out inferior solutions and preserves only those that are "better" than the main population in the objective space but are currently infeasible. These solutions physically represent the active barriers preventing the MP from advancing further.

Step 2: Boundary-Focused Selection. The selection strategy depends on the quantity of high-quality infeasible solutions:

- Case A: Surplus of Infeasible Solutions ($|S_{infeas}| > N$). If the archive is overflowing with infeasible candidates, we prioritize those that define the boundary. We calculate fitness based on negative objective values ($-F(x)$). In the context of minimization, maximizing objectives (via negative sorting) favors solutions with "worse" objective values. Geometric Interpretation: In the infeasible region between the UPF and CPF, solutions with worse objectives are further from the UPF and closer to the feasible boundary (CPF). Thus, negative sorting ensures LI tightly hugs the infeasible side of the CPF.

- Case B: Insufficient Infeasible Solutions ($|S_{infeas}| \leq N$). If there are not enough infeasible solutions to fill the archive, we retain all of them and fill the remaining slots with feasible solutions. Unlike the infeasible case, these feasible solutions are selected based on normal objective values ($F(x)$). Geometric Interpretation: Feasible solutions with better objective values are geometrically closer to the CPF from the feasible side. Including them ensures that LI maintains a dense and continuous representation of the boundary, acting as anchor points for the search.

Through this dual-strategy, LI maintains a set of solutions that effectively outline the boundary of the CPF. Any update to this set signals that the MP has pushed closer to a constraint boundary, triggering the resurrection mechanism.

2) Update AP_i: Algorithm 3 details the update procedure for constraint-specific Auxiliary Populations (AP_i). Given that these populations function primarily as exploratory agents to locate specific boundaries, computational efficiency is prioritized over strict diversity maintenance. Consequently, instead of employing the compu-

---

Algorithm 3: Update Constraint-Specific Population (AP_i)

Input: AP_i (Current Archive), $\mathcal{O}$ (Offspring), N (Total Size), i (Constraint Index), D_i (Direction), MP (Main Population)

Output: AP_i (Updated Archive), F_i (Fitness)

1   $S \leftarrow AP_i \cup \mathcal{O}$;

2   $N_{sub} \leftarrow \lfloor N/4 \rfloor$;   // Target size for this AP_i

    // Part 1: Positive Direction (Search for SCPF)

3   **if** D_i is *Positive* **then**

4     $F_i \leftarrow$ Calculate fitness based on $F(x)$ with CDP under c_i (kNN diversity);

5     $[AP_i, F_i] \leftarrow$ Select top N_{sub} solutions based on F_i;

6   **return** AP_i, F_i;

    // Part 2: Negative Direction (Search for RCPF)

7   $S_{target} \leftarrow \{s \in S \mid c_i(s) \geq 0\}$;   // Violating solutions

8   $S_{reserve} \leftarrow \{s \in S \mid c_i(s) < 0\}$;   // Satisfying solutions

    // Calculate fitness for target solutions

9   $F_{tar} \leftarrow$ Calculate Fitness for S_{target} using $\{F(x), c_i(x)\}$ (kNN diversity);

    // Identify solutions dominated by Main Population

10   $S_{penalized} \leftarrow \{s \in S_{target} \mid \exists m \in MP, m \prec s\}$;

11   $S_{best} \leftarrow S_{target} \setminus S_{penalized}$;

12   **if** $|S_{best}| \leq N_{sub}$ **then**

13     $AP_i \leftarrow S_{best}$;

14     $K \leftarrow N_{sub} - |S_{best}|$;

15   **if** $K > 0$ **then**

      // Fill remaining slots with best penalized solutions

16       $F_{pen} \leftarrow$ Calculate fitness for $S_{penalized}$ using $F(x)$ with CDP under c_i;

17       $AP_i \leftarrow AP_i \cup$ Select top K from $S_{penalized}$ based on F_{pen};

18   **else**

      // Approach RCPF from UPF side (Diversity Priority)

19     $F_{best} \leftarrow$ Calculate fitness for S_{best} based on $-F(x)$ (kNN diversity);

20     $AP_i \leftarrow$ Select top N_{sub} from S_{best} based on F_{best};
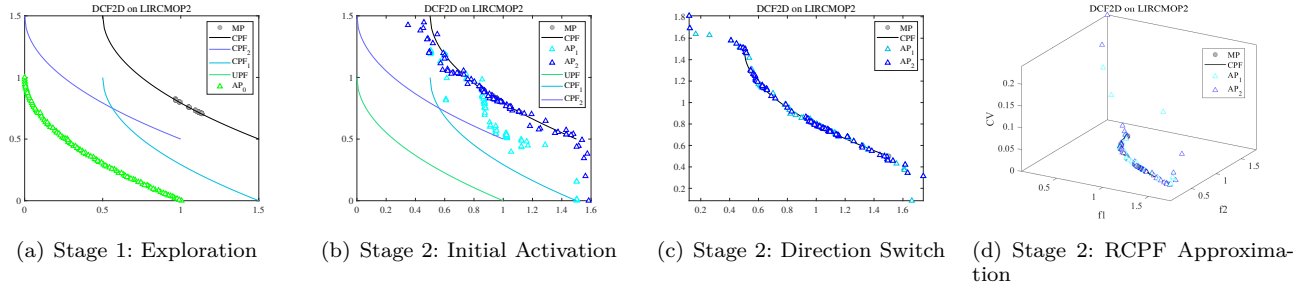
21   **return** AP_i, F_i;

(a) Stage 1: Exploration   (b) Stage 2: Initial Activation   (c) Stage 2: Direction Switch   (d) Stage 2: RCPF Approximation

Fig. 3. Evolutionary process of DCF2D on LIRCMOP2. (a) In Stage 1, AT_0 searches for the UPF. (b) At the beginning of Stage 2, AT_1 and AT_2 are activated in the Positive direction (triangles). (c) Since no feasible solutions are found, the direction switches to Negative, searching for the RCPF (inverted triangles). (d) AT_1 and AT_2 successfully approximate the infeasible boundaries closest to the CPF.

tationally expensive iterative archive truncation operator found in standard SPEA2, this algorithm utilizes a faster selection mechanism based directly on k-Nearest Neighbor (k-NN) density estimation. Although k-NN density provides a slightly less rigorous diversity guarantee than iterative truncation, it offers sufficient distribution pressure for boundary exploration populations while significantly reducing the computational overhead, especially given the reduced population size of $\lfloor N/4 \rfloor$.

When the search direction D_i is Positive, the objective is to converge onto the feasible Pareto front defined by constraint i ($SCPF_i$). The algorithm merges the current population with offspring and applies the Constraint Dominance Principle (CDP), strictly considering only the violation of the specific constraint i. Solutions are then ranked and selected based on their fitness values derived from the k-NN density. This efficient selection mechanism drives the population rapidly towards the feasible region of the specific constraint without the latency introduced by complex diversity maintenance operators.

In the Negative direction, the goal shifts to approximating the Restricted CPF ($RCPF_i$) from the infeasible side. The candidate pool is first filtered to identify target solutions that violate constraint i. Crucially, when calculating the fitness for these target solutions, a multi-objective approach is employed where the constraint violation $c_i(x)$ is treated as an additional objective to be minimized alongside the original objectives. This strategy is designed to prevent the retention of "edge solutions"—individuals that possess superior objective values but suffer from a significantly higher degree of constraint violation (i.e., lying deep in the infeasible region). By optimizing both the objectives and the constraint violation simultaneously, the algorithm forces the population to concentrate on the frontal edge of the CPF boundary rather than drifting towards the sides. Finally, for the selection among high-quality target solutions, fitness is calculated based on negative objective values ($-F(x)$) combined with k-NN density. This ensures that the population tightly hugs the RCPF boundary, efficiently triggering the main population's resurrection mechanism.

C. Analysis of DCF2D on a Running Instance

To provide an intuitive understanding of the proposed mechanism, we analyze the behavior of DCF2D on LIR-CMOP2, a CMOP with two constraints. As illustrated in Fig. 3, the cyan line represents $SCPF_1$, the purple line represents $SCPF_2$, the green line represents the UPF, and the black line represents the final CPF. Notably, in this problem, the final CPF is disjoint from both $SCPF_1$ and $SCPF_2$, corresponding to Situation 4 discussed in Section I (Fig. 1(d)). Therefore, simply searching for the SCPF of these constraints (Positive direction) is insufficient to locate the final CPF.

The evolutionary process proceeds as follows:

1) Stage 1 - Exploration (Fig. 3(a)): AP_0 is utilized to search for the UPF, while the Main Population (MP) searches for the CPF. This broad exploration helps identify the relative position of the UPF.
2) Stage 2 - Activation SCPF Search (Fig. 3(b)): Upon entering Stage 2, the resurrection mechanism identifies Constraints 1 and 2 as potentially useful based on the 2. Consequently, AP_1 and AP_2 are spawned. Initially, their default direction is Positive (searching for SCPFs), indicated by the upward triangles.
3) Stage 2 - Switch to RCPF Search (Fig. 3(c)): As evolution continues, AP_1 and AP_2 fail to locate any feasible solutions in the Positive direction (since the SCPFs are far from the search region or blocked). According to the logic in Case 2 of Algorithm 1, this triggers a direction switch: D_1 and D_2 are set to Negative. The populations now search for $RCPF_1$ and $RCPF_2$ (indicated by inverted triangles).
4) Stage 2 - RCPF Search (Fig. 3(d)): The populations AP_1 and AP_2 now explicitly search for the infeasible boundaries. Corresponding to the negative direction logic in Algorithm 3, the constraint violation $c_i(X)$ is treated as an additional objective to be minimized. As seen in Fig. 3(d), solutions closer to the CPF naturally have smaller constraint violations, while those farther away have larger violations. By optimizing this metric, AP_1 and AP_2 allocate resources to approximate the boundaries of the infeasible regions that define the final CPF (i.e., the RCPF).

The final result (shown in Fig. SF1-a in Supplementary Material) confirms that combining these RCPF approximations allows the Main Population to converge to the true CPF.

D. Experimental Settings

1) Comparative CMOEAs: To comprehensively evaluate the performance of DCF2D, we selected eight state-of-the-art (SOTA) CMOEAs for comparison. These include five constraint decoupling-based methods: C3M [11], MTOTC [13], MSCMO [10], MCCMO [12], and FCDS [14]; and three recent high-performance algorithms: IMTCMO [53], ACREA [24], and APSEA [25]. The selection of decoupling-based methods allows for a direct comparison of the effectiveness of the proposed bidirectional decoupling strategy. IMTCMO, ACREA, and APSEA represent the latest advancements in handling complex constraints.

To ensure a fair comparison while respecting the original design of each algorithm, the reproduction operators are set as follows (same as the original paper): ACREA and APSEA employ Genetic Algorithm (GA) operators (SBX and Polynomial Mutation); MCCMO utilizes GA or Differential Evolution (DE) operators according to CMOPs; and all other algorithms (C3M, MTOTC, MSCMO, FCDS, IMTCMO, and DCF2D) employ DE operators [54]. The population size N is set to 100 for all algorithms across all experiments.

2) Test CMOPs: We conducted experiments on seven benchmark test suites: DASCMOP [55], DOC [35], LIRCMOP [56], MW [57], ZXH_CF [58], CDTLZ [59], and SDC [53]. These suites were selected because they contain multiple constraints and cover all four coupling scenarios discussed in Section I (noting that multiple scenarios may coexist in a single CMOP). Additionally, to validate the performance of DCF2D in practical applications, we tested it on 28 real-world CMOPs (RWMOPs) [60].

- DASCMOP comprises nine difficulty-adjustable CMOPs, allowing users to tune convergence, diversity, and feasibility via three parameters.
- DOC consists of nine CMOPs with constraints in both objective and decision spaces, creating a complex landscape of constraint violation.
- LIRCMOP contains fourteen CMOPs characterized by large infeasible regions that are often independent of the CPF but pose significant obstacles during evolution.
- MW comprises fourteen CMOPs featuring four types of relationships between the UPF and CPF: complete overlap, partial overlap, partial separation, and complete separation.
- ZXH_CF includes sixteen problems with two constraint types: one introduces infeasible barriers guiding the search via correlated variables, while the other defines feasible regions yielding CPFs of varied shapes.

- CDTLZ introduces multiple inequality constraints to divide the feasible domain into non-convex or multi-segment regions.
- SDC imposes constraints directly on decision variables (e.g., distance constraints), testing the algorithm's ability to maintain feasibility in the decision space.
- RWMOP is a suite of real-world problems (CEC 2021) drawn from mechanical design, chemical processes, and synthesis applications.

3) Parameter Settings and Statistical Analysis: To ensure fairness, the termination criterion is defined by the maximum number of Function Evaluations (MaxFE). For the DASCMOP and LIRCMOP suites, MaxFE is set to 300,000, consistent with the standard settings in their respective benchmark papers to allow sufficient convergence. For other test suites, MaxFE is set according to the default configurations in PlatEMO 4.6. All experiments were conducted on a workstation equipped with an Intel Core i7-13700 CPU , 48GB RAM, and Windows 11 OS. To assess the statistical significance of the results, the Wilcoxon rank-sum test with a significance level of 0.05 is employed. In the experimental tables, symbols '+', '−', and '=' indicate that the result of the comparison algorithm is significantly better than, significantly worse than, or statistically similar to that of DCF2D, respectively.

4) Performance Indicator: Given that the true CPF is known for the seven benchmark suites, we employ the Modified Inverted Generational Distance (IGD+) [61] to evaluate performance. Unlike the traditional IGD, IGD+ is weakly Pareto-compliant and provides a more accurate assessment of convergence and diversity by penalizing non-dominated solutions less severely. IGD+ is calculated as follows:

$$IGD^+(P,Q) = \frac{1}{|P|} \sum_{v \in P} \min_{u \in Q} d^+(v,u), \qquad (16)$$

where $d^+(v,u) = \sqrt{\sum_{i=1}^{M}(\max\{f_i(u) - f_i(v), 0\})^2}$. P is a set of reference points uniformly distributed on the true CPF, and Q is the set of non-dominated solutions obtained by the algorithm. The specific IGD+ calculation uses 10,000 reference points via PlatEMO 4.6.

For RWMOPs, where the true CPF is unknown, we use the Hypervolume (HV) [62] metric. HV measures the volume of the objective space dominated by the solution set Q and bounded by a reference point R. A larger HV value indicates better performance in terms of both convergence and diversity. HV is calculated as:

$$HV(Q,R) = L\left(\bigcup_{u \in Q} [f_1(u), r_1] \times \cdots \times [f_M(u), r_M]\right), \qquad (17)$$

where $L(\cdot)$ denotes the Lebesgue measure, and $R = (r_1, r_2, \ldots, r_M)$ is the reference point, set to $[1, 1, \ldots, 1]$ times the nadir point of the combined populations according to the objective dimension M.

TABLE III
Summary of Statistical Significance ($+/-/=$) of IGD+ Values and Average Runtime (Seconds) on Benchmark Suites and Real-world Problems. '+', '−', and '=' indicate that the comparative algorithm is significantly better than, worse than, or similar to DCF2D, respectively (Wilcoxon rank-sum test, $p < 0.05$). Runtime is measured in seconds.

Suite	Metric	Non-Decoupling SOTA			Constraint Decoupling SOTA					Proposed
		ACREA	APSEA	IMTCMO	C3M	FCDS	MCCMO	MSCMO	MTOTC	DCF2D
DASCMOP	Sig. $(+/-/=)$	0/9/0	1/6/2	0/5/4	0/9/0	0/8/1	0/9/0	0/9/0	0/9/0	–
	Runtime (s)	91.23	15.77	57.54	60.30	155.54	77.87	29.71	253.92	72.51
	Avg. Rank	5.33	5.78	2.22	5.78	4.33	6.11	6.89	7.22	1.33
DOC	Sig. $(+/-/=)$	0/9/0	0/9/0	0/3/6	3/5/1	0/8/1	4/5/0	3/5/1	1/3/5	–
	Runtime (s)	96.73	34.34	89.40	69.87	182.22	50.90	60.75	310.44	127.35
	Avg. Rank	9.00	8.11	3.56	3.89	6.44	3.11	3.78	4.67	2.89
CDTLZ	Sig. $(+/-/=)$	5/3/2	4/4/2	0/10/0	0/10/0	0/10/0	0/10/0	0/10/0	0/10/0	–
	Runtime (s)	143.80	42.22	63.48	28.56	326.83	58.12	28.13	143.59	96.97
	Avg. Rank	2.30	3.90	5.20	6.70	3.50	6.40	8.40	6.50	2.10
LIRCMOP	Sig. $(+/-/=)$	6/8/0	0/14/0	3/7/4	0/13/1	3/7/4	2/11/1	0/12/2	3/8/3	–
	Runtime (s)	114.02	17.11	98.10	41.74	220.09	69.88	36.56	231.57	72.42
	Avg. Rank	3.14	9.00	4.36	6.43	4.00	4.64	6.00	4.71	2.71
MW	Sig. $(+/-/=)$	1/11/2	8/5/1	2/9/3	0/14/0	5/6/3	0/14/0	0/14/0	0/13/1	–
	Runtime (s)	111.83	29.02	61.14	22.04	257.73	47.87	20.04	121.16	76.93
	Avg. Rank	5.21	2.21	3.50	6.93	2.71	7.21	7.86	6.79	2.57
SDC	Sig. $(+/-/=)$	0/14/1	1/12/2	7/3/5	3/8/4	3/7/5	2/8/5	3/7/5	2/10/3	–
	Runtime (s)	139.66	29.00	34.51	28.45	177.16	39.17	28.08	91.69	63.42
	Avg. Rank	8.07	5.93	2.33	5.67	4.40	4.93	4.33	6.20	3.20
ZXH_CF	Sig. $(+/-/=)$	10/4/2	7/3/6	0/13/3	1/12/3	1/12/3	0/12/4	0/12/4	0/15/1	–
	Runtime (s)	136.10	39.63	91.81	42.05	247.75	104.35	39.04	427.02	87.89
	Avg. Rank	3.06	3.56	5.00	6.13	3.63	6.63	7.13	6.31	3.56
RWCMOP	Sig. $(+/-/=)$	5/15/2	4/14/8	2/15/11	3/18/7	5/12/11	3/16/9	1/13/11	10/10/8	–
	Runtime (s)	187.73	65.91	195.50	106.81	376.58	237.04	82.41	518.32	108.57
	Avg. Rank	6.21	5.46	4.82	5.71	4.25	5.39	6.21	3.54	3.46
Overall	Total Sig.	27/73/9	25/67/21	14/65/36	10/89/16	17/70/28	11/85/19	7/82/23	16/78/21	–
	Avg. Time	127.64	34.50	86.97	48.05	245.20	86.68	39.88	261.11	86.51
	Avg. Rank	5.29	5.49	3.87	5.90	4.15	5.55	6.33	5.74	2.73

E. Comparison Between DCF2D and State-of-the-art CMOEAs

1) Comparison on test suites CMOPs: Table III (and Table ST1 in the Supplementary Material) presents the comprehensive results of IGD+ values across 87 test instances from seven benchmark suites and 28 real-world problems. The pairwise statistical significance is verified using the Wilcoxon rank-sum test with a 0.05 significance level. To further assess the global ranking differences, the Nemenyi post-hoc test was applied, yielding a Critical Difference (CD) of approximately 1.12 at a significance level of 0.05 As shown in the last row, DCF2D achieves a dominant performance with the best overall average rank of 2.73. Notably, the rank gap between DCF2D and the second-best algorithm, IMTCMO (Rank 3.87), is 1.14, which exceeds the CD threshold, statistically confirming that DCF2D significantly outperforms the nearest competitor. Against decoupling-based methods like C3M, MTOTC, and the recent FCDS, DCF2D demonstrates an overwhelming advantage, achieving significantly better results in the vast majority of cases.

On the DASCMOP suite, DCF2D exhibits exceptional robustness, achieving a remarkable average rank of 1.33. It performs significantly better than all five decoupling-based competitors (C3M, FCDS, MCCMO, MSCMO, MTOTC) on all 9 instances. Compared to the non-decoupling algorithms, DCF2D performs significantly better than ACREA on 9 instances and APSEA on 6 instances. Even against the strong competitor IMTCMO, DCF2D is significantly better on 5 problems and statistically similar on 4. This superiority stems from DCF2D's ability to handle the specific constraint types in DASCMOP. Typically, the second type of constraint is related to the final CPF, while the infeasible region of the first type renders parts of the first type's feasible region infeasible, causing the CPF to differ from the UPF. The third type contains interacting infeasible regions that further block the feasible space. Algorithms like FCDS, which rely on fuzzy dominance without explicit direction, and ACREA, which relies on relaxation, struggle here. DCF2D performs better because it not only considers single constraints from the positive direction but also actively searches from the negative (reverse) direction, allowing the areas blocked by the third type of constraints to be effectively explored.

Regarding the DOC suite, DCF2D achieves the best average rank of 2.89 among all algorithms. It performs significantly better than ACREA and APSEA on all 9 problems, and significantly better than FCDS on 8 problems. Compared to C3M, the performance is mixed, with DCF2D being significantly better on 3 problems and similar on 5. The reason for this strong performance is that although DOC contains many constraints, only a few are actually related to the final CPF, and the optimal

search direction may be bidirectional. Existing methods like MTOTC (significantly worse on 5 problems) treat constraints statically. DCF2D performs better on most DOC problems because it accurately identifies and decouples the critical constraint bundle from two directions to discover the CPF, whereas other algorithms expend excessive evaluation budgets on irrelevant exploration or get stuck in local optima.

In the CDTLZ suite, DCF2D demonstrates high stability and dominance, securing the top rank of 2.10. It performs significantly better than C3M, FCDS, MCCMO, MSCMO, MTOTC, and IMTCMO on all 10 problems. Even against APSEA and ACREA, DCF2D maintains a clear lead. This success is largely due to the discrete feasible regions introduced by CDTLZ's constraints. Unidirectional decoupling methods like C3M and non-decoupling methods like IMTCMO struggle to jump between these disjoint islands. DCF2D handles constraints individually via helper populations, which act as bridges to connect these regions. Moreover, since these problems are prone to falling into local optima, searching from the reverse direction (RCPF) provides an alternative pathway to escape local traps, helping to overcome the local optimum that hinders other algorithms.

On the LIRCMOP suite, characterized by large infeasible regions, DCF2D performs remarkably well, achieving the best average rank of 2.71. It performs significantly better than purely positive-direction search methods like APSEA (on all 14 problems) and C3M (on 13 problems). It also holds a clear advantage over IMTCMO (significantly better on 7 problems, worse on 3). Notably, DCF2D performs significantly better than FCDS on 7 problems. The reason lies in the nature of LIRCMOP, where the CPF is often determined by the boundary of a specific constraint's infeasible region. If an algorithm searches only in the positive direction (like C3M) or relies on global relaxation (like ACREA), it may be blocked by these large infeasible domains. DCF2D's strategy of searching for the RCPF from the negative direction turns these barriers into guides.

On the MW suite, the competition is more intense. DCF2D performs significantly better than decoupling-based methods like C3M (14 problems) and MTOTC (13 problems). However, it performs significantly worse than APSEA on 8 problems. Consequently, APSEA achieves the best rank (2.21), while DCF2D follows with a rank of 2.57. This may be due to the fact that many MW problems feature specific geometric relationships or local optima that favor the Genetic Algorithm operators and adaptive population sizing used in APSEA over the DE operators in DCF2D. Nevertheless, the fact that DCF2D significantly outperforms FCDS (significantly better on 6, similar on 3) and C3M confirms that within the scope of constraint decoupling strategies, DCF2D's bidirectional approach remains superior to unidirectional or fuzzy logic-based decoupling.

Similarly, on the ZXH_CF suite, DCF2D performs significantly better than C3M, FCDS, MCCMO, MSCMO,

and MTOTC on almost all problems (e.g., significantly better than MTOTC on 15 problems). However, it performs significantly worse than ACREA on 10 problems. ACREA achieves the best rank (3.06) on this suite, while DCF2D maintains a competitive rank of 3.56. ZXH_CF involves correlated position and distance variables. ACREA's dual-archive mechanism with adaptive constraint relaxation appears particularly well-suited for traversing these narrow, correlated feasible corridors. In contrast, DCF2D performs better than other decoupling algorithms because its Stage 3 design helps refine the solution distribution.

Finally, on the SDC suite, DCF2D performs significantly worse than IMTCMO on 7 problems and significantly better on 2 problems. From the average values, the gap is not large. IMTCMO ranks first with 2.33, followed by DCF2D with a rank of 3.20. A primary reason is that most constraints in SDC are defined in the decision space (e.g., distance constraints), making their mapping in the objective space extremely complicated and less suitable for objective-space decoupling. However, compared with other decoupling algorithms, DCF2D performs significantly better (e.g., better than MTOTC on 10 problems, better than FCDS on 7 problems). This indicates that while decision-space constraints are challenging, DCF2D's dynamic bidirectional mechanism is more robust than static (MTOTC) or fuzzy (FCDS) decoupling strategies.

We further analyzed the performance of DCF2D relative to the number of constraints (N_{con}) using the average rank of IGD+ values (see Fig. 4). Overall, DCF2D demonstrates exceptional scalability, particularly in environments with a high number of constraints.

In scenarios with low-to-medium constraints ($1 \leq N_{con} \leq 6$), DCF2D maintains a stable performance, consistently fluctuating between Rank 2 and 4. Specifically, at $N_{con} = 3$, DCF2D achieves a high rank (approx. 2.3), outperforming most competitors. A slight performance dip is observed at $N_{con} = 6$ (Rank ≈ 4.3), where static decoupling methods like MCCMO temporarily take the lead. This is likely due to the specific landscape of the DASCMOP or SDC instances with 6 constraints, where the coupling is less dynamic.

However, the most distinct advantage of DCF2D is observed in highly constrained problems ($N_{con} \geq 7$). As shown in Fig. 4, the ranking of DCF2D exhibits a significant upward trend as complexity increases. At $N_{con} = 7, 10,$ and 11, DCF2D achieves the best or near-best rankings (Rank $1 \sim 2$), significantly surpassing the non-decoupling SOTA, IMTCMO. At $N_{con} = 13$, IMTCMO performs slightly better (Rank 1.0) compared to DCF2D (Rank 2.0), largely due to the specific characteristics of the ZXH_CF instances in this group. Notably, in the most complex scenario ($N_{con} = 14$, primarily from the DOC suite), DCF2D rises to the absolute top position (Rank 1.0), while IMTCMO drops to Rank 2.0.

This trend confirms that the proposed bidirectional decoupling strategy effectively mitigates the "curse of dimensionality" in constraint satisfaction. As the number

of constraints grows, the probability of the CPF being defined by complex infeasible boundaries increases, making DCF2D's negative-direction search increasingly critical.
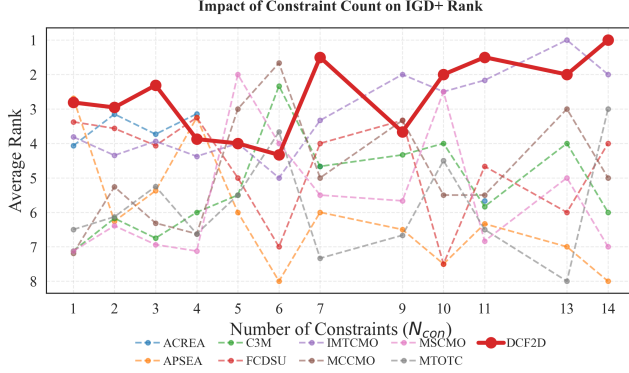

Fig. 4. Impact of constraint count on the average IGD+ rank of DCF2D and comparison algorithms. The y-axis is inverted so that a higher position indicates a better ranking (lower numerical value).

2) *Comparison on real-world CMOPs:* Real-world CMOPs (RWMOPs) are typically characterized by highly nonlinear constraints that create narrow or disjoint feasible regions, posing significant challenges even when the decision space is not excessively large. In these scenarios, DCF2D outperforms the comparison algorithms in terms of both HV values and statistical significance, achieving the best average rank of 3.46. We attribute this superiority primarily to the enhanced diversity maintained by the bi-directional search strategy in Stage 2. By simultaneously approximating the SCPF (from the positive direction) and the RCPF (from the negative direction), DCF2D preserves a diverse set of boundary solutions, preventing the population from becoming trapped in local feasible islands. Furthermore, on specific problems like RWMOP11, DCF2D exhibits superior convergence. From the perspective of constraint decoupling, this is because traditional unidirectional methods often ignore the ICPF. In contrast, DCF2D's unique negative-direction search effectively targets the RCPF, which serves as a critical stepping stone to discover and converge onto the ICPF.

F. Algorithm Complexity Analysis

The computational complexity of DCF2D is primarily governed by the environmental selection mechanism. In Stage 1, the simultaneous execution of the main population (MP) and the exploration population (AP_0), both based on the SPEA2 selection strategy, results in a baseline complexity of $O(2MN^2)$ per generation, where M is the number of objectives and N is the population size.

In Stage 2, while the dynamic constraint decoupling mechanism introduces additional auxiliary tasks, the computational overhead is significantly optimized through two strategic designs. First, the update of the Infeasible Archive (LI) and the MP involves non-dominated sorting, maintaining a complexity of $O(MN^2)$. Second, for

the auxiliary populations (AP_i), we employ a simplified selection mechanism based on k-Nearest Neighbor (k-NN) density estimation, which has a lower complexity of approximately $O(N \log N)$. Furthermore, the population size of each AP_i is reduced to $N/4$. Consequently, in the worst-case scenario where all n_{con} constraints require active auxiliary populations, the per-generation complexity is approximately:

$$O\left(MN^2 + n_{con} \cdot \frac{N}{4} \log \frac{N}{4}\right) \tag{18}$$

This indicates that the marginal cost of adding a helper task is significantly lower than that of the main task. Therefore, DCF2D achieves a comparable or even superior efficiency level relative to existing multi-population methods like C3M [11] and MCCMO [12], which typically scale linearly with $O(n_{con}MN^2)$ due to the maintenance of full-sized populations for each task.
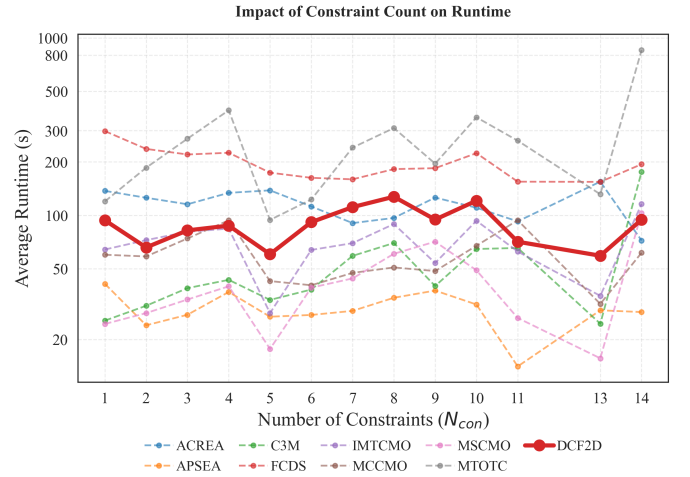

Fig. 5. Impact of the number of constraints (n_{con}) on the average runtime of DCF2D and comparative algorithms. The broken axis is used to accommodate the high runtime of MTOTC and FCDS.

To empirically evaluate the efficiency, we analyzed the relationship between the number of constraints and the actual runtime, as illustrated in Fig. 5. It can be observed that while the runtime of DCF2D generally exhibits an upward trend as the number of constraints (n_{con}) increases, it is not strictly proportional to n_{con}. For instance, the runtime at $n_{con} = 11$ is notably lower than at $n_{con} = 8$ or $n_{con} = 10$. This non-linear behavior validates the efficiency of the proposed dynamic coupling detection mechanism: unlike static decoupling methods (e.g., MTOTC) that process all constraints, DCF2D does not blindly spawn auxiliary populations. Instead, it selectively activates populations only for constraints explicitly identified as shaping the final CPF (either via their SCPF or RCPF). Therefore, for problems where many constraints are redundant or not coupled to the Pareto front, DCF2D avoids unnecessary computational overhead, maintaining a runtime comparable to, and often faster than, other decoupling-based algorithms while

significantly outperforming static methods like MTOTC and FCDS in terms of efficiency.

G. Ablation Study and Parameter Sensitivity

To verify the effectiveness of the proposed bidirectional strategy and the multi-stage framework, we conducted a comprehensive analysis involving algorithm variants and parameter sensitivity. The results are summarized in Table IV.

First, we examine the impact of the search direction strategies. We designed two single-direction variants: DCF2D_POS, which only decouples constraints in the positive direction (approximating SCPF), and DCF2D_NEG, which restricts the search to the negative direction (approximating RCPF). As shown in the top section of Table IV, the standard DCF2D (Bidirectional) achieves the best average ranking of 2.95, significantly outperforming both single-direction variants. These results indicate that searching solely for the feasible boundary or the infeasible boundary is insufficient to capture the complex geometry of diverse CMOPs. The bidirectional mechanism allows the algorithm to adaptively utilize the most informative boundary, ensuring robust convergence.

Second, we investigate the necessity of the final convergence phase (Stage 3) and the sensitivity of its control parameter β. The parameter β determines the fraction of the evaluation budget spent before deactivating auxiliary tasks. As shown in the bottom section of Table IV, when $\beta = 1.0$ (i.e., DCF2D_NoS3 where Stage 3 is removed), the performance drops drastically to the worst rank of 5.33. We attribute this performance degradation, especially on 3-objective CMOPs, to the characteristics of evolutionary environmental selection. Due to the locality of evolutionary operators, offspring solutions tend to be similar to their parents; when auxiliary populations approach the CPF, they continuously generate a large volume of non-dominated solutions. In the environmental selection process, solutions in the same non-dominated level are distinguished by crowding distance (or density). However, as the number of objectives increases, the number of solutions residing in the first non-dominated front increases significantly. Without Stage 3 to stop the influx from auxiliary tasks, the main archive becomes saturated. This forces the selection mechanism to expel solutions that might have a better global distribution in favor of redundant solutions clustered in high-density areas (often provided by auxiliary tasks). Stage 3 mitigates this by halting external injection, allowing the main population to stabilize and refine its global distribution using its own internal diversity maintenance mechanism.

Conversely, when β is too small (e.g., 0.3), the auxiliary tasks are terminated prematurely, failing to fully decouple complex constraints. The algorithm achieves optimal performance at $\beta = 0.9$, suggesting that allocating 90% of the budget to decoupled exploration ensures constraint boundaries are well-identified, while reserving the final 10% for the main population allows for high-precision refinement without interference.

TABLE IV
Average Ranking of Ablation Variants (Search Direction) and Parameter β Sensitivity (Lower Rank indicates Better Performance).

Analysis Category	Variant / Setting	Avg. Rank
Search Direction	DCF2D_POS (Positive Only)	4.60
	DCF2D_NEG (Negative Only)	4.36
	DCF2D (Bidirectional)	2.95
Parameter β	$\beta = 0.3$	3.92
	$\beta = 0.5$	3.20
	$\beta = 0.7$	3.64
	$\beta = 0.9$ (Default)	2.95
	$\beta = 1.0$ (No Stage 3)	5.33

IV. Conclusion

Real-world CMOPs are characterized by multiple constraints exhibiting complex coupling relationships, which pose significant challenges to traditional evolutionary algorithms. Understanding and utilizing these couplings is crucial for efficient optimization. In this paper, we proposed a novel constraint decoupling perspective by introducing the concepts of SCPF, ICPF, and RCPF. We identified that existing decoupling methods primarily focus on the feasible Pareto front of single constraints (SCPF), thereby neglecting the critical role of infeasible boundaries (RCPF) in forming the final CPF. To address this, we proposed the DCF2D algorithm, which employs a bidirectional search strategy. DCF2D operates in three stages: first, exploring the objective space via the UPF to detect constraint couplings; second, spawning auxiliary helper tasks to approximate the CPF from both the positive (SCPF) and negative (RCPF) directions; and finally, concentrating resources on the main population to ensure convergence and diversity. Extensive experiments on seven benchmark suites and 28 real-world CMOPs demonstrate that DCF2D significantly outperforms state-of-the-art CMOEAs, particularly in problems with large infeasible regions and complex constraint interactions.

Despite its promising performance, DCF2D has certain limitations that merit further investigation. First, the current decoupling strategy treats constraints individually. It does not account for composite constraints—scenarios where multiple constraints (e.g., constraint 1 and constraint 2) must be satisfied simultaneously to form a meaningful feasible region, rather than acting independently. In such highly coupled cases, decomposing them one by one may be inefficient. Second, although we employed dynamic resource allocation, maintaining multiple auxiliary populations inevitably incurs additional computational overhead compared to single-population methods.

Future work will focus on two main directions. To address the issue of composite constraints, we aim to develop an automatic constraint grouping mechanism that can identify and bundle interdependent constraints into a single auxiliary task, thereby improving search efficiency. Furthermore, the geometric insights provided by DCF2D offer a strong theoretical foundation for Dynamic

Constrained Multi-objective Optimization Problems (DC-MOPs). In dynamic environments, the relationship between constraints often shifts over time (e.g., a transition from the partial overlap shown in Fig. 1(c) to the complete separation in Fig. 1(d) as infeasible regions expand). Extending the adaptive direction switching mechanism of DCF2D to track these dynamic changes represents a promising avenue for future research.

References

[1] X. Zhou, H. Wang, B. Ding, W. Peng, and R. Wang, "Multi-objective evolutionary computation for topology coverage assessment problem," Knowledge-Based Systems, vol. 177, pp. 1–10, 2019.

[2] X. Zhou, H. Wang, W. Peng, B. Ding, and R. Wang, "Solving multi-scenario cardinality constrained optimization problems via multi-objective evolutionary algorithms," Science China Information Sciences, vol. 62, pp. 1–18, 2019.

[3] X. Xia, Y. Liu, C. Zheng, X. Zhang, Q. Wu, X. Gao, X. Zeng, and Y. Su, "Evolutionary multiobjective molecule optimization in an implicit chemical space," Journal of Chemical Information and Modeling, vol. 64, no. 13, pp. 5161–5174, 2024.

[4] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin, "A coevolutionary framework for constrained multiobjective optimization problems," IEEE Transactions on Evolutionary Computation, vol. 25, no. 1, pp. 102–116, 2020.

[5] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," TIK report, vol. 103, 2001.

[6] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," in Proceedings of the 1999 ACM symposium on Applied computing, 1999, pp. 351–357.

[7] R. Chen, Y.-S. Tsay, and T. Zhang, "A multi-objective optimization strategy for building carbon emission from the whole life cycle perspective," Energy, vol. 262, p. 125373, 2023.

[8] J.-H. Zhu, J.-S. Wang, Y. Zheng, X.-Y. Zhang, X. Liu, X.-T. Wang, and S.-B. Zhang, "Two-stage coevolutionary constrained multi-objective optimization algorithm for solving optimal power flow problems with wind power and facts devices," Renewable Energy, vol. 232, p. 121087, 2024.

[9] F. Ming, W. Gong, H. Zhen, L. Wang, and L. Gao, "Constrained multi-objective optimization evolutionary algorithm for real-world continuous mechanical design problems," Engineering Applications of Artificial Intelligence, vol. 135, p. 108673, 2024.

[10] H. Ma, H. Wei, Y. Tian, R. Cheng, and X. Zhang, "A multi-stage evolutionary algorithm for multi-objective optimization with complex constraints," Information Sciences, vol. 560, 01 2021.

[11] R. Sun, J. Zou, Y. Liu, S. Yang, and J. Zheng, "A multi-stage algorithm for solving multi-objective optimization problems with multi-constraints," IEEE Transactions on Evolutionary Computation, 2022.

[12] J. Zou, R. Sun, Y. Liu, Y. Hu, S. Yang, J. Zheng, and K. Li, "A multi-population evolutionary algorithm using new cooperative mechanism for solving multi-objective problems with multi-constraint," IEEE Transactions on Evolutionary Computation, 2023.

[13] G. Li, Z. Wang, W. Gao, and L. Wang, "Decoupling constraint: Task clone-based multi-tasking optimization for constrained multi-objective optimization," IEEE Transactions on Evolutionary Computation, 2024.

[14] W. Huang, R. Wang, T. Zhang, S. Qi, and L. Wang, "Fuzzy constraint dominance strategy for constrained multiobjective optimization problems with multiple constraints," IEEE/CAA Journal of Automatica Sinica, vol. 12, no. 1, pp. 1–14, 2025.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," IEEE transactions on evolutionary computation, vol. 6, no. 2, pp. 182–197, 2002.

[16] B. Tessema and G. G. Yen, "A self adaptive penalty function based algorithm for constrained optimization," in 2006 IEEE international conference on evolutionary computation. IEEE, 2006, pp. 246–253.

[17] Z. Wu and T. Walski, "Self-adaptive penalty approach compared with other constraint-handling techniques for pipeline optimization," Journal of water resources planning and management, vol. 131, no. 3, pp. 181–192, 2005.

[18] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," IEEE Transactions on evolutionary computation, vol. 4, no. 3, pp. 284–294, 2000.

[19] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," IEEE Transactions on evolutionary computation, vol. 10, no. 6, pp. 658–675, 2006.

[20] F. Ming, W. Gong, L. Wang, and L. Gao, "A constraint-handling technique for decomposition-based constrained many-objective evolutionary algorithms," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 53, no. 12, pp. 7783–7793, 2023.

[21] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, and C. Yue, "An evolutionary multitasking optimization framework for constrained multiobjective optimization problems," IEEE Transactions on Evolutionary Computation, vol. 26, no. 2, pp. 263–277, 2022.

[22] C. He, M. Li, C. Zhang, H. Chen, P. Zhong, Z. Li, and J. Li, "A self-organizing map approach for constrained multi-objective optimization problems," Complex & Intelligent Systems, vol. 8, no. 6, pp. 5355–5375, 2022.

[23] K. Li, R. Chen, G. Fu, and X. Yao, "Two-archive evolutionary algorithm for constrained multiobjective optimization," IEEE Transactions on Evolutionary Computation, vol. 23, no. 2, pp. 303–315, 2019.

[24] R. Wang, W. Huang, W. Li, X. Tang, T. Zhang, and L. Wang, "An adaptive constraint relaxation strategy based coevolutionary algorithm for constrained multi-objective optimization," IEEE Transactions on Emerging Topics in Computational Intelligence, 2025.

[25] Y. Tian, R. Wang, Y. Zhang, and X. Zhang, "Adaptive population sizing for multi-population based constrained multi-objective optimization," Neurocomputing, vol. 621, p. 129296, 2025.

[26] Z.-Z. Liu, B.-C. Wang, and K. Tang, "Handling constrained multiobjective optimization problems via bidirectional coevolution," IEEE Transactions on Cybernetics, vol. 52, no. 10, pp. 10 163–10 176, 2022.

[27] M. Ming, A. Trivedi, R. Wang, D. Srinivasan, and T. Zhang, "A dual-population-based evolutionary algorithm for constrained multiobjective optimization," IEEE Transactions on Evolutionary Computation, vol. 25, no. 4, pp. 739–753, 2021.

[28] J. Wang, G. Liang, and J. Zhang, "Cooperative differential evolution framework for constrained multiobjective optimization," IEEE transactions on cybernetics, vol. 49, no. 6, pp. 2060–2072, 2018.

[29] J. Zhang, J. Cao, F. Zhao, and Z. Chen, "A constrained multi-objective optimization algorithm with two cooperative populations," Memetic Computing, vol. 14, no. 1, pp. 95–113, 2022.

[30] Q. Bao, M. Wang, G. Dai, X. Chen, Z. Song, and S. Li, "A dual-population based bidirectional coevolution algorithm for constrained multi-objective optimization problems," Expert Systems with Applications, vol. 215, p. 119258, 2023.

[31] K. Yang, J. Zheng, J. Zou, F. Yu, and S. Yang, "A dual-population evolutionary algorithm based on adaptive constraint strength for constrained multi-objective optimization," Swarm and Evolutionary Computation, vol. 77, p. 101247, 2023.

[32] Z.-Z. Liu, F. Wu, J. Liu, Y. Qin, and K. Li, "Constrained multiobjective optimization with escape and expansion forces," IEEE Transactions on Evolutionary Computation, 2023.

[33] K. Qiao, K. Yu, C. Yue, B. Qu, M. Liu, and J. Liang, "A cooperative multistep mutation strategy for multiobjective optimization problems with deceptive constraints," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2024.

[34] Z. Liu, F. Han, Q. Ling, H. Han, and J. Jiang, "Constraint-pareto dominance and diversity enhancement strategy based evolutionary algorithm for solving constrained multiobjective optimization problems," IEEE Transactions on Evolutionary Computation, 2025.

[35] Z.-Z. Liu and Y. Wang, "Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces," IEEE Transactions on Evolutionary Computation, vol. 23, no. 5, pp. 870–884, 2019.

[36] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. Yugen, J. Mo, C. Wei, and E. Goodman, "An improved epsilon constraint-handling method in moea/d for cmops with large infeasible regions," Soft Computing, vol. 23, 12 2019.

[37] J. Dong, W. Gong, F. Ming, and L. Wang, "A two-stage evolutionary algorithm based on three indicators for constrained multi-objective optimization," Expert Systems with Applications, vol. 195, p. 116499, 2022.

[38] Y. Tian, Y. Zhang, Y. Su, X. Zhang, K. Tan, and Y. Jin, "Balancing objective optimization and constraint satisfaction in constrained evolutionary multi-objective optimization," IEEE Transactions on Cybernetics, 08 2020.

[39] J. Dong, W. Gong, and F. Ming, "A tri-stage competitive swarm optimizer for constrained multi-objective optimization," Applied Intelligence, vol. 53, no. 7, pp. 7892–7916, 2023.

[40] K. Zhang, Z. Xu, G. G. Yen, and L. Zhang, "Two-stage multi-objective evolution strategy for constrained multi-objective optimization," IEEE Transactions on Evolutionary Computation, 2022.

[41] F. Ming, W. Gong, L. Wang, and L. Gao, "Constrained multi-objective optimization via multitasking and knowledge transfer," IEEE Transactions on Evolutionary Computation, 2022.

[42] J. Zou, R. Sun, S. Yang, and J. Zheng, "A dual-population algorithm based on alternative evolution and degeneration for solving constrained multi-objective optimization problems," Information Sciences, vol. 579, pp. 89–102, 2021.

[43] Y. Zhang, Y. Tian, H. Jiang, X. Zhang, and Y. Jin, "Design and analysis of helper-problem-assisted evolutionary algorithm for constrained multiobjective optimization," Information Sciences, vol. 648, p. 119547, 2023.

[44] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, C. Yue, H. Lin, and K. C. Tan, "Dynamic auxiliary task-based evolutionary multitasking for constrained multi-objective optimization," IEEE Transactions on Evolutionary Computation, 2022.

[45] F. Ming, W. Gong, L. Wang, and C. Lu, "A tri-population based co-evolutionary framework for constrained multi-objective optimization problems," Swarm and Evolutionary Computation, vol. 70, p. 101055, 2022.

[46] J. Zhou, Y. Zhang, J. Wang, and P. N. Suganthan, "Localized constrained-domination principle for constrained multiobjective optimization," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 54, no. 3, pp. 1376–1387, 2023.

[47] L. Si, X. Zhang, Y. Zhang, Y. Tian, and S. Yang, "Reinforcement learning-assisted multi-stage evolutionary constrained multi-objective optimization," ACM Transactions on Evolutionary Learning, 2025.

[48] F. Ming, W. Gong, and L. Gao, "Adaptive auxiliary task selection for multitasking-assisted constrained multi-objective optimization [feature]," IEEE Computational Intelligence Magazine, vol. 18, no. 2, pp. 18–30, 2023.

[49] K. Qiao, J. Liang, K. Yu, M. Wang, B. Qu, C. Yue, and Y. Guo, "A self-adaptive evolutionary multi-task based constrained multi-objective evolutionary algorithm," IEEE Transactions on Emerging Topics in Computational Intelligence, 2023.

[50] M. Ming, R. Wang, H. Ishibuchi, and T. Zhang, "A novel dual-stage dual-population evolutionary algorithm for constrained multiobjective optimization," IEEE Transactions on Evolutionary Computation, vol. 26, no. 5, pp. 1129–1143, 2021.

[51] X. Ban, J. Liang, K. Qiao, K. Yu, Y. Wang, J. Peng, and B. Qu, "A decision variables classification-based evolutionary algorithm for constrained multi-objective optimization problems," IEEE/CAA Journal of Automatica Sinica, 2025.

[52] B. Xu, Y. Zheng, W. Li, X. Gao, D. Gong, J. He, and Z. Fan, "Handling multiobjective optimization problems with complex constraints: A constraints grouping-based approach," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2025.

[53] K. Qiao, J. Liang, K. Yu, C. Yue, H. Lin, D. Zhang, and B. Qu, "Evolutionary constrained multiobjective optimization: Scalable high-dimensional constraint benchmarks and algorithm," IEEE Transactions on Evolutionary Computation, 2023.

[54] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," IEEE Transactions on Evolutionary Computation, vol. 15, pp. 4 – 31, 03 2011.

[55] Z. Fan, W. Li, X. Cai, H. Li, Q. Zhang, K. Deb, and E. Goodman, "Difficulty adjustable and scalable constrained multi-objective test problem toolkit," Evolutionary Computation, vol. 28, pp. 1–28, 05 2019.

[56] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. Yugen, J. Mo, C. Wei, and E. Goodman, "An improved epsilon constraint-handling method in moea/d for cmops with large infeasible regions," Soft Computing, vol. 23, 12 2019.

[57] Z. Ma and Y. Wang, "Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons," IEEE Transactions on Evolutionary Computation, vol. PP, pp. 1–1, 02 2019.

[58] Y. Zhou, Y. Xiang, and X. He, "Constrained multiobjective optimization: Test problem construction and performance evaluations," IEEE Transactions on Evolutionary Computation, vol. 25, no. 1, pp. 172–186, 2020.

[59] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach," IEEE Transactions on evolutionary computation, vol. 18, no. 4, pp. 602–622, 2013.

[60] A. Kumar, G. Wu, M. Z. Ali, Q. Luo, R. Mallipeddi, P. N. Suganthan, and S. Das, "A benchmark-suite of real-world constrained multi-objective optimization problems and some baseline results," Swarm and Evolutionary Computation, vol. 67, p. 100961, 2021.

[61] P. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," IEEE Transactions on Evolutionary Computation, vol. 7, no. 2, pp. 174–188, 2003.

[62] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," IEEE Transactions on Evolutionary Computation, vol. 10, pp. 29 – 38, 03 2006.