Handling Constrained Multi-objective Optimization By Ignoring Constraints and Using Two Evolutionary Frameworks

Yi Xiang, Xiaowei Yang, Han Huang, *Senior Member, IEEE* and Jiahai Wang *Senior Member, IEEE*

***Abstract*—Constrained multi-objective optimization problems exist widely in real-world applications, and they involve a simultaneous optimization of multiple and often conflicting objectives subject to several equality and/or inequality constraints. To deal with these problems, a crucial issue is how to handle constraints effectively. This paper proposes a simple yet effective constrained decomposition-based multi-objective evolutionary algorithm. In the proposal, the evolutionary process is divided into two stages in which constraints are handled differently. In the first stage, constraints are totally ignored and the population is pulled toward the unconstrained Pareto-optimal front (PF) by optimizing objectives only. This can help the proposed algorithm handle well problems with the following features, i.e., the constrained PF has an intersection with the unconstrained counterpart, and there are infeasible regions blocking the way of convergence. In the second stage, with the purpose of approximating the constrained PF well, constraint satisfaction is emphasized over objective minimization. Moreover, different evolutionary frameworks are adopted in the two stages to promote the performance of the algorithm as much as possible. The proposed algorithm is comprehensively compared with several state-of-the-art algorithms on 39 problems (with 266 test instances in total), including one real-world problem (with 36 instances) in search-based software engineering. As shown by the experimental results, the new algorithm performs best on the majority of these problems, particularly on those with the aforementioned features. In summary, the suggested algorithm provides an effective way of handling constrained multi-objective optimization problems.**

***Index Terms*—Constrained multi-objective optimization; decomposition-based algorithms; constraint handling techniques; two frameworks**

I. Introduction

CONSTRAINED multi-objective optimization problems (CMOPs) exist widely in real-world applications, such as the optimal scheduling of energy storage systems [1], the optimization of biped robot gaits [2] and reliability-based optimization for crashworthy structures [3]. This type of problems involves a simultaneous optimization of multiple and often conflicting objectives, and a set of equality and/or inequality constraints. Without loss of generality, a CMOP can be defined as follows.

$$\begin{aligned} \text{Minimize} \quad & \boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, p \\ & h_j(\boldsymbol{x}) = 0, j = 1, \ldots, q \\ & \boldsymbol{x} \in \Omega \subset \mathbb{R}^n, \end{aligned} \tag{1}$$

where $\boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T \in \mathbb{R}^m$ and $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T \in \Omega$ (the decision space) are the objective vector and decision vector, respectively. In Formula (1), $g_i(\boldsymbol{x}) \leq 0$ defines the i-th of the p inequality constraints, and $h_j(\boldsymbol{x}) = 0$ defines the j-th of the q equality constraints. According to [4], CMOPs with more than three objectives are known as constrained many-objective optimization problems (CMaOPs).

From the perspective of decision makers, feasible solutions are naturally emphasized over infeasible ones. A solution to CMOP (1) is said to be feasible provided that it meets all the equality and inequality constraints. If any constraint is violated, this solution is infeasible. To distinguish between solutions, one applicable and often used principle is Pareto-dominance. The $\boldsymbol{x}_1$ is said to Pareto-dominate $\boldsymbol{x}_2$ (written as $\boldsymbol{x}_1 \prec \boldsymbol{x}_2$) if and only if $f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2)$ for all $i = 1, \ldots, m$ and $f_j(\boldsymbol{x}_1) < f_j(\boldsymbol{x}_2)$ for at least one $j \in \{1, \ldots, m\}$. For a solution $\boldsymbol{x}^*$, if there exist no other solutions dominating it, then $\boldsymbol{x}^*$ is called a Pareto-optimal solution. All the Pareto-optimal solutions constitute the Pareto-optimal set (PS), and the Pareto-optimal front (PF) is defined as the image of the PS in the objective space.

The CMOPs can be divided into three types based on the relationship between the constrained PF (CPF) and the unconstrained PF (UPF). For Type-I CMOPs, as shown in Fig. 1 (a), the CPF is exactly the same as the UPF. For Type-II problems, the CPF is or contains a part of the UPF. As illustrated in Fig. 1 (b) and (c), the CPF consists of a part of the unconstrained front, while in Fig. 1 (d) the CPF contains not only a part of the UPF but also a part of the feasible boundary. Finally, as illustrated in Fig. 1 (e), the CPF for Type-III CMOPs has no intersection with the UPF. Mathematically, if the intersection of CPF and UPF is A, then $|A| = |UPF|$, $0 < |A| < |UPF|$ and $|A| = 0$ for Type-I, Type-II and Type-III CMOPs, respectively. Moreover, as shown in Fig. 2, each type of CMOPs may have infeasible regions blocking the way of converging towards the CPF [4], [5]. To emphasize this feature, CMOPs with infeasible barriers are called Type-I′, Type-II′ and Type-III′ problems in this paper.

(*Corresponding author: Han Huang*)

Y. Xiang, X. Yang and H. Huang are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: xiangyi@scut.edu.cn; xwyang@scut.edu.cn; hhan@scut.edu.cn).

J Wang is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China.

Digital Object Identifier xxxxxxxxxxxx

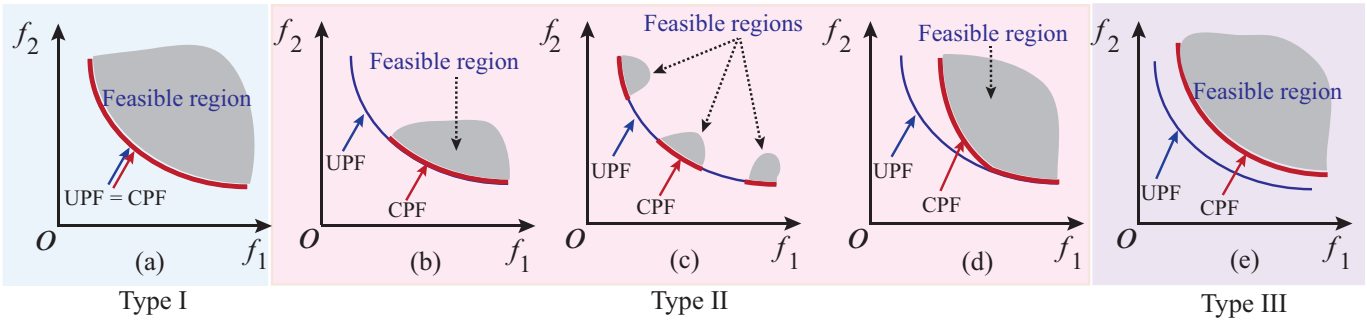


Fig. 1. Illustration of three types of CMOPs. For Type-I CMOPs, $CPF = UPF$. For Type-II CMOPs, $CPF \bigcap UPF = A$, where A is a set and $0 < |A| < |UPF|$. For Type-III CMOPs, $CPF \bigcap UPF = \emptyset$.

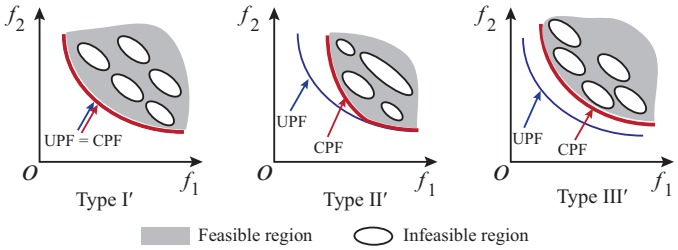


Fig. 2. In Type-I′, Type-II′ and Type-III′ problems, there are infeasible regions blocking the way of converging towards the CPF.

Despite that most, if not all, real-world problems have various constraints in nature, it is quite surprising that studies on constrained multi-objective evolutionary algorithms (CMOEAs) have not attracted enough attention from the evolutionary multi-objective optimization (EMO) community [5], [6]. Instead, much more attention has been paid to the solving of unconstrained multi-objective optimization problems (MOPs). Indeed, a number of unconstrained multi-objective evolutionary algorithms (MOEAs) have been proposed during the last two decades [7]–[11]. In these algorithms, a key issue that should be well addressed is how to maintain a balance between convergence and diversity. This promotes researchers to propose various selection schemes in different genres of MOEAs [9]. For example, in Pareto-dominance-based algorithms, e.g., NSGA-II [12], GrEA [13], VaEA [14], NSGA-III [15], solutions in the first several non-dominated layers are selected in priority to improve convergence, and the remaining solutions are selected based on density information to maintain diversity. In decomposition-based algorithms, e.g., MOEA/D [16], RVEA [17], MOEA/DD [18], the convergence is driven by optimizing scalarization functions, and the diversity is maintained by specifying a set of uniformly distributed weight/reference vectors. In indicator-based algorithms, such as IBEA [19] and HypE [20], the performance is promoted by maximizing the hypervolume (HV) metric [21], which measures convergence and diversity in a simultaneous way.

Generally speaking, CMOPs are much more difficult than their unconstrained counterparts, due to that another key issue—how to handle constraints—should be also properly addressed. In fact, there have already been some representative constraint handling techniques (CHTs), including constraint-domination principle (CDP) [12], penalty functions [22], stochastic ranking (SR) [23], ε constrained method [24] (called ε for brevity), methods based on multi-objective optimization [25], and hybrid methods [26]. Although most of the above CHTs were originally proposed for constrained single-objective optimization problems (CSOPs), they have been integrated into MOEAs to handle CMOPs as well, leading to some representative CMOEAs. Like the taxonomy adopted in the unconstrained scenario, CMOEAs can be divided into three mainstreams according to the selection mechanism used, i.e., Pareto-dominance-based, decomposition-based and indicator-based algorithms.

Typical Pareto-dominance-based CMOEAs include NSGA-II-CDP [12], C-NSGA-III [27], IDEA [28], C-VaEA [29], the CMOEA proposed by Woldesenbet et al. [6], etc. Although all the above algorithms use the Pareto dominance to rank solutions, they are distinct with respect to the way in which constraints are handled. More specifically, constraints in both NSGA-II-CDP and C-NSGA-III are handled by CDP, which prefers consistently a feasible solution over an infeasible solution. In both IDEA [28] and C-VaEA [29], a small percentage of infeasible solutions are retained to exploit information carried by them. This allows the algorithms to approach the constrained boundaries from both the feasible and infeasible sides of the search space. In [6], constraints are handled based on an adaptive penalty function and a distance measure. Since decomposition-based algorithms convert an MOP into a number of scalar sub-problems, most CHTs can be directly or at least easily integrated into this framework to tackle CMOPs. This makes decomposition-based CMOEAs popular in the field. Representative algorithms include ε-based methods (e.g., CMOEA/D [30], MOEA/D-IEpsilon [31], PPS-MOEA/D [32]), CDP-based method (e.g., CMOEA/D-CDP [33]), SR-based methods (e.g., CMOEA/D-SR [33], MOEA/D-ASR [34]), and extended decomposition-based MOEAs (e.g., C-MOEA/DD [18], C-RVEA [17]). Notice that indicator-based CMOEAs have been rarely studied in the EMO community [35]. Recently, Liu et al. [36] made the first study on combining indicator-based MOEAs with different CHTs, including feasibility rule (FR) [12], SR and ε. Results suggest that the performance of this kind of CMOEAs is sensitive to both indicator-based MOEAs and the adopted CHTs.

By Ignoring Constraints and using two evolutionary frameworks, we propose in this paper a simple yet effec-

tive constrained decomposition-based algorithm (called CIC-MOEA/D). In the algorithm, the evolutionary process is automatically divided into two stages, in which different evolutionary frameworks are adopted and constraints are also handled in different ways. In the first stage, constraints are totally ignored, and a clustering-based ranking framework is developed to balance convergence and diversity (details are given in Section III-A). In the second stage, a steady-state framework as in MOEA/D [16] is applied as usual, and a new solution is allowed to replace only a small number of neighboring solutions (to ensure diversity) [37]. With the purpose of approximating the CPF well, constraints are emphasized over objectives by employing a CHT that considers in order three criteria when comparing solutions, i.e., constraint violations (CV), Pareto-dominance and scalarization functions. Note that the third criterion makes this technique also effective in dealing with CMaOPs.

The rationales of ignoring constraints in the first stage are twofold. First, as shown in Fig. 1, it would be salutary to ignore constraints in searching for CPF of Type-I and Type-II problems. In fact, these types of CMOPs can be totally or partially "solved" because the whole CPF or parts of it can be well approximated by ignoring constraints. Second, for Type-I′, Type-II′ and Type-III′ problems, by ignoring constraints, the working population can easily get across obstacles caused by infeasible regions, being likely to escape from traps which are feasible regions surrounded by infeasible blocks [5], [32]. It must be noted that the recently proposed push and pull search (PPS) [32] applied a similar idea. That is, constraints are not taken into consideration in the early stage. However, our method differs from PPS in the frameworks used, and particularly in the search in the second stage[1]. As for the reasons why two evolutionary frameworks should be adopted simultaneously, a recent study in [38] has revealed that it is not expected that a single framework can perform fairly well on different CMOPs. It will be also demonstrated by the experimental results in Section VI-B that the simultaneous use of the two frameworks indeed performs significantly better than the use of only one framework.

Main contributions of the paper are summarized as follows.

- By analyzing the relationship between CPF and UPF, we demonstrate that ignoring constraints is an effective way of handling some specific CMOPs, like Type-I and Type-II problems, and those with infeasible barriers.
- A simple yet effective CMOEA is proposed. The main characteristic of the new algorithm is that the evolutionary process is automatically divided into two stages, in each of which a different evolutionary framework and a different CHT are adopted.
- We demonstrate the effectiveness and efficiency of the proposed algorithm by conducting a comprehensively comparative experimental study, in which CIC-MOEA/D is compared with several state-of-the-art CMOEAs on a number of both artificial and practical CMOPs, including MW test problems [38], DC-DTLZ test problems [5], DAS-CMOPs/DAS-CMaOPs [4] and the real-world constrained optimal software product selection (OSPS) problem [39]. All these problems have been proposed recently and can be classified based on our taxonomy.

The remainder of the paper is organized as follows. Section II reviews related work on CHTs. In Section III, we give details of the proposed CIC-MOEA/D. Section IV specifies the experimental setup, followed by results and discussions presented in Section V. Subsequently, investigations on the effect of ignoring constraints and using two frameworks are given in Section VI. Finally, Section VII concludes the paper and outlines possible directions for future studies.

II. Related work

The CDP, first proposed in the NSGA-II paper [12], and also adopted in C-NSGA-III [27] and the MOEA/D framework [33], may be the simplest and most widely used CHT for CMOPs. This principle prefers feasible solutions to infeasible ones. Such a preference drives the population to feasibility before improving the objectives [28], and hence may encounter difficulties in getting across obstacles caused by infeasible regions.

Another common approach to handle constraints is to use penalties. Penalty-function-based techniques normally convert CMOPs into unconstrained MOPs by adding a penalty term to the objectives [22], [32]. In these techniques, the penalty factor plays an important role in maintaining the balance between minimizing the objectives and satisfying the constraints. However, the tuning of this factor is not always easy because it is often problem-dependent [38]. This limitation promotes researchers to develop self-adaptive penalty (SP) functions [40], [41], in which penalty factors are adjusted adaptively on the basis of the feedback taken from the search process. To handle CMOPs, Woldesenbet et al. [6] proposed to modify each objective by using a distance measure, and an SP function in which two penalties are involved. The first one is based on objective functions and the second is based on the constraint violation. The balance between the two penalties is adaptively controlled by the ratio of feasible individuals presented in the current population.

In SR [23], for any two solutions, they are compared either according to the objective function with the probability P_f (a user-defined parameter), or according to the constraint violations with the probability $1 - P_f$. Although SR was originally proposed for CSOPs, it can be applied to handle CMOPs as well. In [42], a new CHT with infeasible elitists preservation and SR-based selection is proposed to handle constraints in multi-objective optimization. Given that the Pareto dominance may lead to incomparable solutions, the new method assigns each solution a fitness value based on its ranking in the nondominated sorting and its crowding distance. In [33], the extended version of SR was implemented for the first time in the MOEA/D framework [16]. To solve CMOPs effectively, Ying et al. [34] proposed an adaptive SR mechanism in MOEA/D. In this mechanism, the probability P_f is adaptively adjusted by borrowing the idea of metropolis acceptance criterion.

[1]More discussions on the differences between CIC-MOEA/D and PPS are provided at the end of Section III-F after giving technical details of CIC-MOEA/D.

In ε constrained method [24], the constraint violations are relaxed by the ε-level. Solutions with constraint violations less than ε are treated as feasible ones, and they are compared according to the objective functions. In [30]–[32], [43], this CHT was adopted in the framework of MOEA/D to deal with CMOPs. Since MOPs in MOEA/D are transformed into a number of scalar sub-problems using weight vectors, the ε constrained method can be directly applied. However, finding an optimal (or near-optimal) value for ε is not at all trivial. Therefore, the ε level in most cases is set dynamically during the evolutionary process [31], [32], [43].

By converting constraints into one or more extra objectives, a constrained problem is reformulated as an unconstrained MOP, which can be solved by MOEAs [44]. Typical work using this method includes Cai and Wang's method [25], the infeasibility driven evolutionary algorithm [28], a general framework of dynamic CMOEAs for constrained optimization [45], and the tri-goal evolution framework for CMaOPs [46], etc. Inversely, a CMOP can be handled by first transforming a CMOP into a CSOP to find the promising feasible area, and then by applying a specific CMOEA to obtain the final solutions [47].

Constraints can be handled by combining several popular CHTs together in either different evolutionary stages or different sub-populations. For example, the adaptive tradeoff model (ATM), proposed by Wang et al. [26], divides the search process into three scenarios. If solutions in the current population are all infeasible, a multi-objective approach is adopted to deal with constraints. If the population contains both feasible and infeasible solutions, ATM uses a penalty function to select solutions for the next generation. In case of all feasible solutions, the comparison is performed based on entirely the objective values. Despite that ATM was originally proposed for CSOPs, Ma et al. [38] have recently investigated its performance on CMOPs, showing that ATM could obtain competitive results as well, particularly on those with narrow feasible regions. The ensemble of constraint handling methods (ECHM), suggested by Qu et al. [48], was used to tackle CMOPs. In ECHM, three CHTs are involved, including an SP function [6], the superiority of feasible solutions [49] and the ε constrained method [24]. Note that the ECHM uses three sub-populations, each of which employs a different constraint-handling method.

Recently, ignoring constraints has become a feasible way of handling CMOPs. In PPS-MOEA/D [32], the search process is divided into push and pull stages. In the push stage, constraints are entirely ignored, and the population is pushed toward the UPF as close as possible by optimising objectives only. In the pull stage, the population is pulled back to the CPF using an improved ε constraint-handling approach. In C-TAEA [5], two collaborative archives are maintained simultaneously. The convergence-oriented archive (CA) is the driving force to push the population toward the PF, and diversity-oriented archive (DA) focuses on maintaining the population diversity. One of the main characteristics of the update mechanism for DA is that constraint violation is not taken into consideration.

III. The Proposed CIC-MOEA/D

The pseudo-code of the proposed CIC-MOEA/D is outlined in Algorithm 1. As shown, CIC-MOEA/D has two algorithm-level parameters: the neighbor size (T) and the parameter used in the switch between two stages (gap)[2], in addition to two common parameters, i.e., the population size (N) and the number of the maximum generations (G). The algorithm begins with the initialization of N weight vectors $\Lambda = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N\}$ (Line 1), the determination of neighbor structure for sub-problems (Line 2) and the initialization of N random solutions $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ (Line 3). Note that the i-th weight vector $\boldsymbol{\lambda}_i$ in Λ is denoted as $\boldsymbol{\lambda}_i = (\lambda_{i_1}, \ldots, \lambda_{i_j}, \ldots, \lambda_{i_m})$, where $\lambda_{i_j} \geq 0$ and $\sum_{j=1}^{m} \lambda_{i_j} = 1$. In the algorithm, we record the best feasible solution for each sub-problem, and these solutions are stored in the set P'. According to Line 5, members in P are used to initialize P'. Specifically, the i-th member in P' (denoted by $\boldsymbol{x}'_i$) is initialized as $\boldsymbol{x}_i$ if $\boldsymbol{x}_i$ is feasible, and NULL otherwise. Subsequently, as shown in Line 6, the variable $stage$, indicating the current stage, is initialized to 1.

According to Lines 10-25, the evolutionary process is then divided into two stages, where the population is evolved by the first framework ($evolution_{F_1}$) and the second framework ($evolution_{F_2}$), respectively. Notice that constraints are not taken into account in the first stage. During the search, non-dominated feasible solutions are stored in an external archive A (being empty initially), which is updated periodically in the second stage for every gap generations. Finally, after ranking A, which is updated in Line 27, by using the constrained non-dominated sorting [27], the algorithm returns the first layer L_1. Specifically, L_1 contains either all the non-dominated feasible solutions in case that feasible solutions appear in A, or the solution(s) with the smallest constraint violation if all the solutions are infeasible.

In the following subsections, we will give details on the main algorithmic components.

A. The first evolutionary framework

In the first stage, the primary goal is to aid the population in getting across infeasible barriers (if any), and to pull the population towards the UPF. Therefore, constraints are not taken into consideration in the adopted evolutionary framework (i.e., $evolution_{F_1}$ (Λ, P) in Algorithm 2). Instead, this framework puts more emphasise on balancing convergence and diversity. To this end, a clustering-based ranking method is developed to select N promising solutions for the next generation. Based on the order in which the solutions are selected, they are grouped into different layers. Each solution has an attribute "*rank*", indicating which layer it belongs to. This attribute can aid in the mating selection, where solutions with small "rank" values are preferred.

As shown in Line 1 of Algorithm 2, an offspring population Q is generated from the current population P. Details on the reproduction are as follows: for each sub-problem i, a candidate set D is set either to $B(i)$ (with the probability δ, a parameter), or to $\{1, \ldots, N\}$ (with the probability $1 - \delta$),

[2]The parameter gap serves also as the archive update period.

Algorithm 1 The framework of CIC-MOEA/D

Input: population size (N), number of maximum generations (G), neighbor size (T) and the parameter used in the switch between two stages (gap)
Output: Final archived solutions

1: Initialize N weight vectors: $\Lambda = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N\}$
2: For each $i \in \{1, \ldots, N\}$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\boldsymbol{\lambda}_{i_1}, \ldots, \boldsymbol{\lambda}_{i_T}$ are T closest weight vectors to $\boldsymbol{\lambda}_i$
3: Initialize N random solutions: $P \leftarrow \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
4: Compute objectives and evaluate constraints for $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$
5: Initialize feasible solution set $P' = \{\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_N\}$ using members in P
6: $stage \leftarrow 1$ // $stage \in \{1, 2\}$ indicates the stages
7: $g \leftarrow 1$ // g is the current generation
8: $A \leftarrow \emptyset$ // The archive A is empty
9: **while** $g \leq G$ **do**
10: **if** $stage = 1$ **then**
11: $P \leftarrow evolution_{F_1}\ (\Lambda, P)$ // Evolve P using the first framework
12: $g++$
13: $trials \leftarrow updateFeasibleSet\ (P', P)$ // The $trails$ is the number of unsuccessful trials when updating the ideal point for feasible solutions
14: **if** $(trials > gap \times N)\ ||\ (g > G/2)$ **then**
15: $stage \leftarrow 2$
16: $P \leftarrow reconstruct\ (P', P)$ // Reconstruct the population P
17: **end if**
18: **else**
19: $P \leftarrow evolution_{F_2}\ (P)$ // Evolve P using the second framework
20: $g++$
21: // Update archive A for every gap generations
22: **if** $g\ \%\ gap = 0$ **then**
23: $A \leftarrow updateArchive\ (A \cup P)$
24: **end if**
25: **end if**
26: **end while**
27: $A \leftarrow updateArchive\ (A \cup P)$
28: Perform the constrained non-dominated sorting [27] to A, and identify the first layer L_1
29: **return** L_1

Algorithm 2 $P \leftarrow evolution_{F_1}\ (\Lambda, P)$

Input: the set of weight vectors Λ and the population P
Output: the updated population P

1: $Q \leftarrow reproduction\ (P)$
2: $S \leftarrow P \cup Q$ // $|S| = 2N$
3: $S \leftarrow normalize\ (S)$
4: $\boldsymbol{z}^{**} \leftarrow (-\epsilon, \ldots, -\epsilon)$ // Set the reference point
5: $r \leftarrow 1$
6: $k \leftarrow 0$ // The number of selected solutions
7: **while** $k < N$ **do**
8: $\{C_1, \ldots, C_N\} \leftarrow clustering\ (\Lambda, S)$
9: **for** each $i \in \{1, \ldots, N\}$ **do**
10: **if** $|C_i| \geq 2$ **then**
11: $\boldsymbol{y} \leftarrow \arg\min_{\boldsymbol{x} \in C_i} g^{pbi}(\boldsymbol{x} | \boldsymbol{z}^{**}, \boldsymbol{\lambda}_i)$
12: $\boldsymbol{y}.rank \leftarrow r$
13: $\boldsymbol{x}_i \leftarrow \boldsymbol{y}$ // Update the i-th solution in P using $\boldsymbol{y}$
14: $k++$ // The counter k is increased by one
15: **If** $k = N$, **then return** P
16: $S \leftarrow S \setminus \{\boldsymbol{y}\}$ // Remove $\boldsymbol{y}$ from S
17: $\Lambda \leftarrow \Lambda \setminus \{\boldsymbol{\lambda}_i\}$ // Remove $\boldsymbol{\lambda}_i$ from Λ
18: **end if**
19: **end for**
20: $r++$
21: **end while**
22: **return** P

from which two different elements k_1 and k_2 can be chosen. We then apply a binary selection to $\boldsymbol{x}_{k_1}$ and $\boldsymbol{x}_{k_1}$, and the one with a smaller "rank" wins out. Notice that a random selection can be performed in case that both solutions have the same "rank" value. In a similar way, we can choose another parent $\boldsymbol{x}_l$. After crossover, the two parents reproduce two offspring, from which we randomly select the final one (to be mutated and evaluated subsequently). Since only one child is generated each time in the above procedure, the set Q contains exactly N offspring.

According to Line 2 of Algorithm 2, the parent and offspring populations are merged into a single set S, and members in this set are suggested to be normalized (Line 3). Specifically, for each $\boldsymbol{x} \in S$, the i-th objective value $f_i(\boldsymbol{x})$ is normalized to $\widetilde{f}_i(\boldsymbol{x})$ as follows.

$$\widetilde{f}_i(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x}) - z_i^*}{z_i^{max} - z_i^*} \tag{2}$$

where z_i^* is the best objective value for the i-th objective found during the search, and z_i^{max} is the worst objective value for the i-th objective among members in S. In fact, the point $\boldsymbol{z}^* = (z_1^*, \ldots, z_m^*)$ is an estimation to the ideal point. After normalisation, $\boldsymbol{z}^*$ should be $(0, \ldots, 0)$. Following the practice in [50], [51], the reference point $\boldsymbol{z}^{**} = (z_1^{**}, \ldots, z_m^{**})$, used to calculate scalarization functions, can be more optimistic than the normalised $\boldsymbol{z}^*$. Therefore, as shown in Line 4, $\boldsymbol{z}^{**}$ is set to $(-\epsilon, \ldots, -\epsilon)$, where ϵ is a positive number and takes 0.1 in this paper.

Lines 7-21 in Algorithm 2 depict the clustering-based ranking method. First, individuals in S are classified into clusters using the function $clustering\ (\Lambda, S)$ (Line 8). More precisely, we calculate the perpendicular distance (in the objective space) from each solution $\boldsymbol{x}$ to each of the search directions (determined by weight vectors), and associate this solution with the weight vector to which $\boldsymbol{x}$ has the shortest perpendicular distance. As shown in Fig. 3, for example, solutions 1, 9 and 10 are associated with $\boldsymbol{\lambda}_1$, forming the first cluster $C_1 = \{1, 9, 10\}$. Analogously, $C_2 = \{8\}$, $C_3 = \{2, 3, 7\}$, $C_4 = \{6\}$ and $C_5 = \{4, 5\}$.

Second, the solution with the best $g^{pbi}(\boldsymbol{x} | \boldsymbol{z}^{**}, \boldsymbol{\lambda}_i)$ is selected within each cluster that contains at least two members, and is then used to replace the old solution $\boldsymbol{x}_i$ (Line 13).

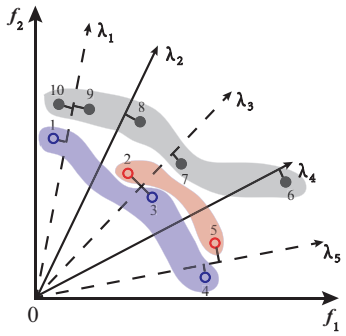


Fig. 3. Illustration of the clustering-based ranking method. The first layer $L_1 = \{1, 3, 4\}$ and the second layer $L_2 = \{2, 5\}$. For solutions 6-10, they are not chosen in the ranking process.

The $g^{pbi}(\boldsymbol{x}|\boldsymbol{z}^{**}, \boldsymbol{\lambda}_i)$ is the penalty-based boundary intersection (PBI) function proposed in [16], and is in the following form.

$$g^{pbi}(\boldsymbol{x}|\boldsymbol{z}^{**}, \boldsymbol{\lambda}_i) = d_1 + \theta d_2, \tag{3}$$

where θ is a user defined penalty parameter; d_1 and d_2 are defined as

$$d_1 = \frac{(\widetilde{\boldsymbol{f}}(\boldsymbol{x}) - \boldsymbol{z}^{**})^T \boldsymbol{\lambda}_i}{\|\boldsymbol{\lambda}_i\|}, \ d_2 = \left\| \widetilde{\boldsymbol{f}}(\boldsymbol{x}) - (\boldsymbol{z}^{**} + d_1 \frac{\boldsymbol{\lambda}_i}{\|\boldsymbol{\lambda}_i\|}) \right\|. \tag{4}$$

These solutions, with the "rank" being r, constitute the r-th layer L_r. For example, the solutions 1, 3 and 4 are selected within C_1, C_3 and C_5, respectively. Therefore, $L_1 = \{1, 3, 4\}$. Notice that, to introduce competition, we only consider clusters with at least two members. As a result, those with less than two members are not taken into account in the current iteration. For example, as shown in Fig. 3, both C_2 and C_4 have only one associated solution, and therefore they are not considered currently.

Third, the selected solutions are removed from S (Line 16). Similarly, the corresponding weight vectors are also removed from Λ (Line 17). In Fig. 3, for example, S and Λ become $\{2, 5, 6, 7, 8, 9, 10\}$ and $\{\boldsymbol{\lambda}_2, \boldsymbol{\lambda}_4\}$, respectively.

Finally, repeat the above steps until N solutions are selected. Taking Fig. 3 as an example again, in the second-round of clustering, solutions 2 and 8-10 are associated with $\boldsymbol{\lambda}_2$, and solutions 5-7 are associated with $\boldsymbol{\lambda}_4$. Since both the two new clusters have more than one solution, we can identify the best solution within each of the clusters to form the second layer $L_2 = \{2, 5\}$, with the assumption that the solutions 2 and 5 are selected from C_2 and C_4, respectively[3].

At the end of this section, we make the following remarks on the first framework.

- Since the clustering is not performed in a one-off manner (and may be repetitively executed if necessary), solutions are not restricted to a fixed cluster during the ranking process, thus enabling a comprehensive evaluation of solutions under different weight vectors. In Fig. 3, for example, the solution 5, belonging to C_5 in the first-round clustering, is re-assigned to C_4 in the second round. Clearly, this solution is not the best under $\boldsymbol{\lambda}_5$, but it indeed performs best under $\boldsymbol{\lambda}_4$ given that the solution 6 is far away from the PF.
- In the first framework, both diversity and convergence are considered. The diversity is promoted by clustering while the convergence is guaranteed by the PBI scalarization function. The balance between them is controlled by the dynamic clustering strategy. As discussed early in Section I, using the first framework alone is able to handle Type-I and Type-II CMOPs/CMaOPs.

[3]The solution 5, rather than solution 6, is selected within C_4, because we assume that solution 6 is far away from the true front.

B. Update feasible solution set and switch between two stages

For each sub-problem i, we record two best solutions: One is the current best solution $\boldsymbol{x}_i \in P$, and the other is the current best feasible solution $\boldsymbol{x}'_i \in P'$. After P is evolved by using the first framework (Line 11 in Algorithm 1), members in P are utilized to update P' via the function $updateFeasibleSet$, which returns $trails$ — the number of unsuccessful trials when updating the ideal point for feasible solutions (Line 13 in Algorithm 1). To be specific, $\boldsymbol{x}'_i$ is replaced by $\boldsymbol{x}_i$ if and only if one of the following conditions is true.

- Solution $\boldsymbol{x}_i$ is feasible and solution $\boldsymbol{x}'_i$ is NULL.
- Solution $\boldsymbol{x}_i$ is feasible, and $\boldsymbol{x}_i$ Pareto-dominates solution $\boldsymbol{x}'_i$.
- Solution $\boldsymbol{x}_i$ is feasible and is also non-dominated with $\boldsymbol{x}'_i$, but $g^{pbi}(\boldsymbol{x}_i|\boldsymbol{z}^{**}, \boldsymbol{\lambda}_i) < g^{pbi}(\boldsymbol{x}'_i|\boldsymbol{z}^{**}, \boldsymbol{\lambda}_i)$.

If $\boldsymbol{x}'_i$ is replaced by $\boldsymbol{x}_i$, then $\boldsymbol{x}_i$ is used to update the ideal point for feasible solutions. If any dimension of this point is updated, then the variable $trails$ is reset to 0. Otherwise, it is increased by one. The role of the variable $trails$ is to determine the switch between the two stages. If $trials > gap \times N$, as shown in Line 14 in Algorithm 1, the process enters into the second stage by assigning 2 to the variable $stage$. Note that the computational resources allocated to the first stage is at most $G/2$. That is to say, the evolution is switched from the first stage to the second stage if the ideal point for feasible solutions has not been updated for $gap \times N$ times, or the current number of generations (i.e., g) exceeds $G/2$.

Before the second stage starts, the current population is reconstructed based on P' and P (Line 16 in Algorithm 1). This construction is quite simple. To be specific, we scan each member $\boldsymbol{x}_i \in P$ and replace it with $\boldsymbol{x}'_i \in P'$, if $\boldsymbol{x}_i$ is infeasible and $\boldsymbol{x}'_i$ is not NULL. The purpose of reconstruction is to conduct the subsequent search from the current best feasible solutions, rather than from the current best solutions because some of them, if not all, may be infeasible.

C. The second evolutionary framework

To approximate well the CPF, constraints are emphasized over objectives in the second-stage evolution. In this stage, we adopt a steady-state evolutionary framework, which integrates a normalization procedure and a $better$ function that compares solutions in the presence of constraints. To begin with, we give the definition of constraint violation (CV) [12], [15] that is widely used to measure the degree of constraints violated

by a solution.

$$CV(\boldsymbol{x}) = \sum_{i=1}^{p} \max\{g_i(\boldsymbol{x}), 0\} + \sum_{j=1}^{q} \max\{|h_j(\boldsymbol{x})| - \delta, 0\}, \tag{5}$$

where δ is a very small positive value (e.g., 10^{-4}) [38] adopted to relax the equality constraints. Apparently, the CV value is equal to 0 for feasible solutions, and larger than 0 for infeasible ones.

As shown in Line 1 of Algorithm 3, $evolution_{F_2}\ (P)$ starts by normalizing the parent population P using the same method as in the first framework. The function $normalize\ (P)$ returns the estimated ideal point ($\boldsymbol{z}^*$) and the max point of P (denoted by $\boldsymbol{z}^{max}$). After this, the reference point $\boldsymbol{z}^{**}$ is set to $(-\epsilon, \ldots, -\epsilon)$. Subsequently, for each sub-problem i, a new solution $\boldsymbol{y}$ is generated by performing in order the mating selection, crossover and mutation (Lines 4-12). According to Line 14, after the evaluation of objectives and constraints, the new solution $\boldsymbol{y}$ is normalized using $\boldsymbol{z}^*$ and $\boldsymbol{z}^{max}$ that are obtained in Line 1. The normalized objectives are then adopted to update $\boldsymbol{z}^{**}$: for each $j = 1, \ldots, m$, if $\tilde{f}_j(\boldsymbol{y}) < z_j^{**}$ is true, then z_j^{**} is replaced by $\tilde{f}_j(\boldsymbol{y})$ (Line 15).

Algorithm 3 $P \leftarrow evolution_{F_2}\ (P)$

Input: the population P
Output: the updated population P

1: $\{\boldsymbol{z}^*, \boldsymbol{z}^{max}\} \leftarrow normalize\ (P)$
2: $\boldsymbol{z}^{**} \leftarrow (-\epsilon, \ldots, -\epsilon)$
3: **for** each $i \in \{1, \ldots, N\}$ **do**
4: **if** $rnd < \delta$ **then**
5: $D = B(i)$
6: **else**
7: $D = \{1, \ldots, N\}$
8: **end if**
9: Randomly select two indexes k and l ($k \neq l$) from D
10: $\{\boldsymbol{y}_1, \boldsymbol{y}_2\} \leftarrow crossover(\boldsymbol{x}_k, \boldsymbol{x}_l)$
11: $\boldsymbol{y} \leftarrow$ a random selection between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$
12: Apply mutation to $\boldsymbol{y}$
13: Compute objectives and evaluate constraints for $\boldsymbol{y}$
14: $\tilde{f}_j(\boldsymbol{y}) = \frac{f_j(\boldsymbol{y}) - z_j^*}{z_j^{max} - z_j^*}$, $j = 1, \ldots, m$
15: Update $\boldsymbol{z}^{**}$: For $j = 1, \ldots, m$, if $\tilde{f}_j(\boldsymbol{y}) < z_j^{**}$, set $z_j^{**} \leftarrow \tilde{f}_j(\boldsymbol{y})$
16: /* Update sub-problems */
17: $times \leftarrow 0$
18: **for** each $q \in D$ **do**
19: **if** $better(\boldsymbol{x}_q, \boldsymbol{y}, \boldsymbol{z}^{**})$ **then**
20: Replace $\boldsymbol{x}_q$ with $\boldsymbol{y}$
21: $times++$
22: **end if**
23: **If** $times \geq n_r$, **then break**
24: **end for**
25: **end for**
26: **return** P

The newly generated $\boldsymbol{y}$ is employed to update at most n_r (a control parameter) sub-problems, and this is achieved by utilizing a variable named $times$. According to Line 23, the procedure breaks in the *for* loop in case that $times \geq n_r$. For each q in D, being either $B(i)$ or $\{1, \ldots, N\}$ (see Lines 4-8), the procedure checks whether $\boldsymbol{y}$ is better than $\boldsymbol{x}_q$ or not. If $\boldsymbol{y}$ is better, then $\boldsymbol{x}_q$ is replaced by $\boldsymbol{y}$. At the same time, $times$ is increased by one. It should be mentioned that, to introduce randomness, we do not follow a fixed order (from the first element to the last one) when scanning the set D in Line 18. Instead, members in D are visited in a random way such that each one has an equal chance to be reached. In this framework, the convergence is promoted by minimizing PBI scalarization, and the diversity is maintained by specifying a set of weight vectors. The balance between them is handled by restricting the number of sub-problems to be updated by a new solution [37].

Algorithm 4 $better(\boldsymbol{x}_q, \boldsymbol{y}, \boldsymbol{z}^{**})$

Input: $\boldsymbol{x}_q$ and $\boldsymbol{y}$
Output: True or False

if $CV(\boldsymbol{y}) < CV(\boldsymbol{x}_q)$ **then**
  return True
else if $CV(\boldsymbol{y}) = CV(\boldsymbol{x}_q)$ **then**
  if $(\boldsymbol{y} \prec \boldsymbol{x}_q)$ **then**
    return True
  else if $(\boldsymbol{y} \not\prec \boldsymbol{x}_q) \wedge (\boldsymbol{x}_q \not\prec \boldsymbol{y})$ **then**
    if $g^{pbi}(\boldsymbol{y}|\boldsymbol{z}^{**}, \boldsymbol{\lambda}_q) < g^{pbi}(\boldsymbol{x}_q|\boldsymbol{z}^{**}, \boldsymbol{\lambda}_q)$ **then**
      return True
    end if
  end if
end if
return False

The last point is related to the $better$ function, which is detailed in Algorithm 4. The $\boldsymbol{y}$ is said to be better than $\boldsymbol{x}_q$ if and only if one of the following conditions is true.

- $\boldsymbol{y}$ has a smaller CV than $\boldsymbol{x}_q$;
- Both $\boldsymbol{y}$ and $\boldsymbol{x}_q$ have the same CV, but $\boldsymbol{y}$ dominates $\boldsymbol{x}_q$ (i.e., $\boldsymbol{y} \prec \boldsymbol{x}_q$);
- Both $\boldsymbol{y}$ and $\boldsymbol{x}_q$ have the same CV, and they are also non-dominated with each other, but $g^{pbi}(\boldsymbol{y}|\boldsymbol{z}^{**}, \boldsymbol{\lambda}_q) < g^{pbi}(\boldsymbol{x}_q|\boldsymbol{z}^{**}, \boldsymbol{\lambda}_q)$.

Clearly, the above comparison follows a constraint-first and objective-second principle, emphasizing first the solutions violating less constraints over those violating more, and second the solutions dominating others over those being dominated. Finally, those with small PBI values are preferred in case that the solutions are incomparable regarding both CV values and the Pareto-dominance relation. It should be mentioned that this constraint handling technique shares a similarity with CDP. That is, both of them compare solutions using CV values first and Pareto-dominance second. However, the new constraint handling technique adopts another criterion, i.e., PBI scalarization function, making this technique more suitable than CDP in handling CMaOPs. As discussed elsewhere [9], [15], [52], the Pareto dominance can lead to insufficient selection pressure as the number of objectives increases. In contrast, scalarization functions compare individuals using a scalar value, being effective in distinguishing between solutions even

in the context of many-objective optimization [15].

D. Update archive

In the second-stage evolution, as shown in Algorithm 1, the external archive A is updated for every gap generations. In this paper, we adopt the procedure introduced in [53] to maintain the archive A. In fact, this procedure was originally designed for the environmental selection in the unconstrained optimization, aiming at choosing N solutions for the next generation from the mixed population with $2N$ solutions. Since the task is similar in maintaining the archive (selecting also N solutions from $A \cup P$ with $2N$ individuals), this procedure can be automatically transferable in this context. Different from the environmental selection performed at each generation, however, the maintenance of archive is executed periodically. Due to space limit, we give a brief introduction to this procedure in Section I in the supplement. Interested readers can find more details in the original study [53].

E. Time complexity

The $clustering$ operation in Line 8 of Algorithm 2 requires $O(mN^2)$ computations, and the for loop from Line 9 to Line 19 requires at most $O(N^2)$ computations. Given that $O(mN^2) > O(N^2)$, the time complexity of the interior of the $while$ loop in Algorithm 2 is $O(mN^2)$. Since the $while$ loop is repeated at most N times. Hence, the worst-case time complexity of Algorithm 2 is $O(mN^3)$. For the second framework (Algorithm 3), the worst time complexity is $O(mN^2)$ in the case in which $|D| = N$. Considering all the above, the worst-case time complexity of CIC-MOEA/D is $O(mN^3)$. In practice, however, CIC-MOEA/D runs much faster than expected. As will be shown in Sections V-A and VI-C, CIC-MOEA/D runs faster than C-NSGA-III [15], C-MOEA/DD [18] and C-TAEA [5]. The following are some possible reasons. First and most importantly, the theoretical time complexity of the first framework (Algorithm 2) is obtained in the worst case in which each solution forms an independent layer. However, this scenario rarely occurs in practice. It is common that multiple solutions belong to a same layer. Hence, the $clustering$ operation in Algorithm 2 is repeated much less than N times. Naturally, this improves the running speed significantly. Second, according to Line 14 in Algorithm 1, the first framework is applied at most $G/2$ generations. This further reduces the runtime consumed.

F. Some discussions on CIC-MOEA/D

- The proposed CIC-MOEA/D share some similarities with C-MOEA/DD. For example, both of them estimate density by associating solutions with weight vectors, and delete/select solutions according to the PBI function. Moreover, solutions are ranked into different layers in both algorithms. However, CIC-MOEA/D is different from C-MOEA/DD in the following aspects. First, C-MOEA/DD divides the population into different layers according to the Pareto dominance relation, whilst CIC-MOEA/D ranks solutions based on the order in which solutions are selected according to PBI function values. Second, when associating solutions with weight vectors, C-MOEA/DD uses angles whereas CIC-MOEA/D uses perpendicular distances.
- Both CIC-MOEA/D and PPS-MOEA/D ignore constraints in the first stage. However, the working principles of the two algorithms are quite different in the second stage. Specifically, as illustrated in Fig. 4, PPS-MOEA/D should at first push the population towards the UPF as close as possible, and then gradually pull it back from UPF to CPF by using the ε constraint handling strategy. When the gap between UPF and CPF is large, however, the pull stage can be time-consuming, and wastes computational resources. In CIC-MOEA/D, the search in the second stage starts with the best feasible solutions recorded during the search in the first stage. This in general makes CIC-MOEA/D more efficient in approximating the CPF in the above scenario. Moreover, the CHTs adopted in the two algorithms are different in the second stage. Finally, each of the two stages in CIC-MOEA/D, as mentioned previously, uses a different evolutionary framework, being different from PPS-MOEA/D in which only one framework is adopted in both stages. The advantages of using two frameworks will be experimentally investigated in Section VI-B.
- We must admit that some techniqes/strategies used in CIC-MOEA/D are inspired by ideas adopted previously. Nevertheless, as will be demonstrated in the following section, the combination of these simple techniqes/strategies can be surprisingly effective in handling most of the existing CMOPs, both artificial and practical. Moreover, the proposed CIC-MOEA/D is computationally efficient (in terms of the actual execution time), being faster than some state-of-the-art CMOEAs, e.g., C-NSGA-III [27], C-TAEA [5].

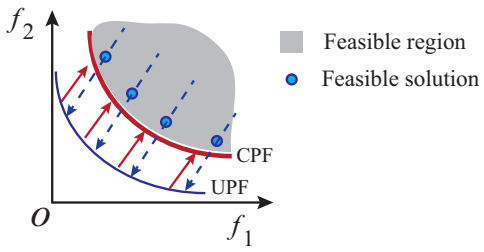


Fig. 4. Illustration of the difference between CIC-MOEA/D and PPS-MOEA/D. In the second stage, the search in PPS-MOEA/D starts from the UPF, while that in CIC-MOEA/D starts from the best feasible solutions ever recorded.

IV. Experimental Setup

A. Test suites

To perform empirical evaluations, we choose four suites of CMOPs/CMaOPs, i.e., the MW test problems proposed by Ma and Wang [38], the DC-DTLZ test problems proposed by Li et al. [5], the DAS-CMOPs/DAS-CMaOPs test problems proposed by Fan et al. [4], and the real-world constrained

OSPS problem proposed by Xiang et al. [39]. All the four test suites have been proposed recently, and each of them has its own features. For example, the MW test suite, consisting of 14 problems, covers diverse characteristics extracted from real-world CMOPs, such as small feasibility ratio, sufficient nonlinearity of constraints [38]. The DC-DTLZ test suite, developed on the basis of DTLZ problems [54], is characterized by constraints acting on the decision space [5]. The most important feature of DAS-CMOPs/DAS-CMaOPs is that the difficulty of the problems can be adjusted using a triplet (η, ζ, γ), where the three parameters specify the difficulty level of diversity-hardness, feasibility-hardness and convergence-hardness, respectively [4]. Finally, the constrained OSPS is a real-world problem in search-based software engineering, with the features of being large-scale and discrete. More details on these test suites can be found in Section II-A in the online supplement. Notice that, apart from CMOPs with 2 or 3 objectives, we also take into account CMaOPs with 4, 5 and 10 objectives. This setting allows us to investigate the scalability of the algorithms with respect to the number of objectives.

B. Peer algorithms and performance metrics

To carry out performance comparisons, we compare the proposed CIC-MOEA/D with six state-of-the-art algorithms, i.e., PPS-MOEA/D [32], C-NSGA-III [27], C-MOEA/D [27], NSGA-II-CDP [12], C-MOEA/DD [18] and C-TAEA [5]. The recently proposed PPS-MOEA/D is an instantiation of the PPS framework embedded in MOEA/D, and constraints in this algorithm are handled by an improved ε method. The C-NSGA-III is the constrained version of NSGA-III [15], and it uses CDP to sort a population into different nondomination layers. Similarly, NSGA-II-CDP also handles constraints via CDP. In C-MOEA/D, a modification of MOEA/D-DE [37], feasibility information, in addition to aggregation function values, is adopted to compare solutions to emphasize feasible and small-CV solutions. The C-MOEA/DD combines Pareto dominance and decomposition into a single paradigm, and feasible solutions are in general emphasized over infeasible ones. However, if an infeasible solution is associated with an isolated subregion, then it will survive without reservation so as to improve population diversity. Finally, the C-TAEA is a tailored algorithm for CMOPs, in which two archives are maintained simultaneously. Note that constraints are not considered when updating the diversity-oriented archive[4].

Each algorithm is independently run 30 times in each test instance, and the medians of performance metrics are used to evaluate and compare algorithms. The performance metrics used in this study are HV [21] and IGD+ [55]. According to [56], both metrics are able to simultaneously measure convergence and diversity. It should be noted that a larger HV value indicates a better solution set. In contrast, a smaller IGD+ value is desired. A brief introduction to the two performance metrics can be found in Section II-B in the online supplement.

[4]The proposed CIC-MOEA/D is independently compared with C-TAEA in Section VI-C due to their similarity.

C. Experimental settings

Due to space limit, experimental settings used in our study are summarized in Section II-C in the online supplement. These settings include the population size N, the termination conditions, reproduction operators along with corresponding parameter settings, and control parameters in peer algorithms.

V. Results and Discussions

In this section, we give computational results on MW test problems [38] (Section V-A), DC-DTLZ test problems [5] (Section V-B), DAS-CMOPs/DAS-CMaOPs [4] (Section V-C), and the real-world constrained OSPS problem [39] (Section V-D). Moreover, some necessary discussions are also presented.

A. Results on MW test problems

Table S-5 in the supplement summarizes the medians of the HV results obtained by each algorithm on the MW test problems. In that table, as well as other related tables, the best and the second-best results for each test problem are shaded with a dark gray and a light gray background, respectively. Moreover, the Wilcoxon's rank sum test [57], at a 0.05 significance level, is performed for each pairwise comparison between CIC-MOEA/D and each of the algorithms on each test problem. The statistical results are represented by three symbols: •, ○ and ‡, indicating that CIC-MOEA/D performs significantly better than, significantly worse than and statistically equivalently to the compared algorithms, respectively.

As seen in Table S-5 in the supplement, CIC-MOEA/D obtains 12 best/second-best results on the MW test suite, outperforming dramatically all the others. This conclusion is further confirmed by the statistical results, summarized in Table I, showing that CIC-MOEA/D is significantly better than its competitors on at least $10/14 \approx 71\%$ and at most $13/14 \approx 93\%$ test problems. Inversely, CIC-MOEA/D is inferior to all the other algorithms, except for PPS-MOEA/D, on at most one test problem. Compared with PPS-MOEA/D, CIC-MOEA/D presents significantly worse performance on only two problems. According to Table S-6 in the supplement, behaviors of the algorithms regarding the IGD+ results are consistent with those regarding HV. This well demonstrates that CIC-MOEA/D is the best-performing algorithm on the MW test problems.

TABLE I
SUMMARY OF COMPARISONS REGARDING HV ON MW TEST PROBLEMS.

CIC-MOEA/D v.s.	•	‡	○
PPS-MOEA/D	10	2	2
C-NSGA-III	12	2	0
C-MOEA/D	13	0	1
NSGA-II-CDP	10	3	1
C-MOEA/DD	13	0	1

For each type of problems, we count the number of test problems on which CIC-MOEA/D obtains the best or second-best HV results, and then calculate the percentages by dividing this number by the total number of problems in each

type. After calculation, we find that CIC-MOEA/D performs best or second best on 100% Type-I problems, 88% Type-II problems, and 67% Type-III problems, indicating that CIC-MOEA/D is particularly suitable for solving Type-I and Type-II problems. Recall that we have asserted in Section I that, by ignoring constraints, CIC-MOEA/D would perform effectively on problems whose CPF contains the entire or part of the UPF. This assertion is well validated by the experimental results presented in this section. In fact, the CPFs of Type-I and Type-II MW problems come either entirely or partly from the unconstrained counterparts [38]. Results show that CIC-MOEA/D can indeed perform well on these problems, particularly on Type-I problems whose CPF is the same as the UPF.

To compare the computational efficiency, we show in Fig. 5 the logarithm of the runtime on all the MW test problems. As seen, CIC-MOEA/D runs slower than C-MOEA/D and NSGA-II-CDP, but faster than C-MOEA/DD and C-NSGA-III. It is observed that, on the 2-objective test problems, the speed of CIC-MOEA/D is very close to that of PPS-MOEA/D. On the 3-objective MW4, MW8 and MW14, however, CIC-MOEA/D runs obviously faster than PPS-MOEA/D. Clearly, CIC-MOEA/D has a moderate running speed among all the algorithms considered here. Without introducing much computational burden, the proposed CIC-MOEA/D can achieve high performance on these problems. This feature makes CIC-MOEA/D a very promising tool for handling CMOPs.

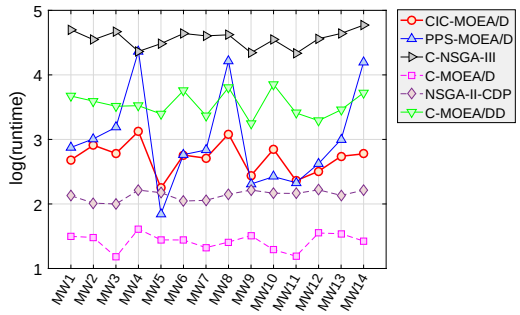


Fig. 5. Runtime comparison on MW test problems.

At the end of this section, we investigate the scalability of the algorithms with respect to the number of objectives by extending MW4, MW8 and MW14 to CMaOPs. The HV results are given in Table S-7 in the supplement, where we can find that CIC-MOEA/D performs best or second best in all the test instances, except for the 5-objective MW8. Being a highly competitive algorithm to CIC-MOEA/D, C-MOEA/DD obtains the best/second-best HV in four test instances. Compared with PPS-MOEA/D, C-MOEA/D and NSGA-II-CDP, CIC-MOEA/D performs significantly better in all the 6 test instances. In comparison with C-NSGA-III, CIC-MOEA/D shows worse performance in only one test instance, i.e., the 5-objective MW8. According to Table S-7 in the supplement, CIC-MOEA/D is the most effective algorithm on the Type-I problems, obtaining the best HV results in all the four test instances. On Type-II problems, for which the best algorithm is C-MOEA/DD, the proposed algorithm performs competitively. We remind the readers that, due to page limit, visualized comparisons on the MW test problems are presented in Figs. S-1 and S-2 in the supplement.

B. Results on DC-DTLZ test problems

According to our taxonomy, the DC-DTLZ test problems can be classified into three types: I, II and II′. The HV results on these problems are listed in Table S-8 in the supplement. Considering the number of the best/second-best results, CIC-MOEA/D is the best-performing algorithm, followed by PPS-MOEA/D and C-MOEA/DD. It is clear from that table that CIC-MOEA/D performs fairly well on all the three types of problems. This is consistent with our expectation. In fact, as discussed in Section I, the three types of problems poss features which can be effectively handled by mechanisms adopted in CIC-MOEA/D. These features include that the CPF has an intersection with UPF, and that there are infeasible regions blocking the way of convergence. For example, DC2-DTLZ1 and DC2-DTLZ3 belong to Type I. However, the feasible region of the two problems is very narrow, and the CV value of a solution does not monotonically decrease when it converges toward the CPF. Moreover, there exist several local optima of the CV function [5]. The above characteristics make the two problems extremely difficult. As shown in Fig. S-3 (a) in the supplement, some algorithms under investigation fail to return any feasible solution (e.g., C-MOEA/DD), and some get stuck in a locally optimal front (i.e., C-NSGA-III, C-MOEA/D and NSGA-II-CDP). Note that, like CIC-MOEA/D, PPS-MOEA/D can also escape from local optima, but its solution set is worse than that of CIC-MOEA/D concerning both convergence and diversity. We are aware that only CIC-MOEA/D and PPS-MOEA/D are able to jump over local optima on DC2-DTLZ1 and DC2-DTLZ3. The reason is that both algorithms do not consider CV values in the early stage of the evolutionary process, and this will make the algorithms immune to local optima of the CV function.

For the 3-objective DC1-DTLZ1, as shown in Fig. S-3 (b) in the supplement, the solutions obtained by CIC-MOEA/D, NSGA-II-CDP and C-MOEA/DD can cover all the disconnected segments. In contrast, at least one segment is missed by the remaining three algorithms (i.e., PPS-MOEA/D, C-NSGA-III and C-MOEA/D). Compared with the solutions of NSGA-II-CDP, those of CIC-MOEA/D are distributed more widely and more uniformly. In addition, it can be observed that C-MOEA/DD returns much less final solutions than CIC-MOEA/D does. Finally, by observing Fig. S-3 (c) in the supplement, we find that only CIC-MOEA/D can yield (nearly) converged solutions on the 3-objective DC3-DTLZ3, while the solutions of other algorithms are all stuck at locally optimal front. According to [5], the feasible region of DC3-DTLZ3 is disconnected. In other words, there are many infeasible segments blocking the way of converging towards the CPF. As discussed in Section I, ignoring constraints as done in CIC-MOEA/D can easily overcome these infeasible barriers. On the contrary, feasibility-driven algorithms, e.g., C-NSGA-III, C-MOEA/D and NSGA-II-CDP, may struggle in converging into the true front primarily because the working population

can hardly jump out from a feasible region to an infeasible one. Notice that constraints are also ignored in the first stage in PPS-MOEA/D, the algorithm, however, cannot well handle DC3-DTLZ3 either. One of the possible explanations is that the population has not been pushed onto or near to the UPF in the push stage where constraints are not considered, thus making it difficult to pull the population back to the CPF in the subsequent pull stage.

C. Results on DAS-CMOPs and DAS-CMaOPs problems

According to the taxonomy adopted in this paper, the DAS-CMOPs and DAS-CMaOPs with convergence-, diversity- and feasibility-hardness are Type-I′/-II′[5], Type-II and Type-III problems, respectively. Since the main characteristic of the two test sets is the difficulty adjustability, we choose a difficulty triplet with moderate hardness for each type. More specifically, the difficulty triplet is $(0, 0, 0.5)$, $(0.5, 0, 0)$ and $(0, 0.5, 0)$ for Type I′/II′, Type II and Type III, respectively. The HV and IGD+ results on DAS-CMOP test problems are available in Tables S-9 and S-10 in the supplement, and the summary of the statistical test results is given in Table II. As seen, CIC-MOEA/D performs significantly better than the peer algorithms in at least 74% and 67% test instances regarding HV and IGD+, respectively. In addition, the CIC-MOEA/D outperforms obviously other algorithms in at most 89% test instances concerning both performance metrics. Clearly, CIC-MOEA/D obtains the best overall performance on DAS-CMOPs. For visualized comparisons on this test suite, please refer to Fig. S-4 in the supplement.

TABLE II
SUMMARY OF COMPARISONS REGARDING HV AND IGD+ ON DAS-CMOPS.

CIC-MOEA/D v.s.	HV	IGD+
	•/ ‡ /○	•/ ‡ /○
PPS-MOEA/D	74%/11%/15%	78%/15%/ 7%
C-NSGA-III	78%/ 4%/19%	67%/15%/19%
C-MOEA/D	85%/ 4%/11%	81%/ 7%/11%
NSGA-II-CDP	85%/ 4%/11%	85%/11%/ 4%
C-MOEA/DD	89%/ 4%/ 7%	89%/ 4%/ 7%

We are particularly interested in comparing the algorithms for each type separately. Fig. 6 (a) presents the percentage of each type of DAS-CMOPs on which the best HV results are obtained by CIC-MOEA/D, PPS-MOEA/D and C-NSGA-III, the three most competitive algorithms on this test suite. It can be found that CIC-MOEA/D overwhelmingly outperforms PPS-MOEA/D and C-NSGA-III on Type-I′/-II′ and Type-II problems, obtaining the best HV results on 78% problems for both types. On Type-III problems, CIC-MOEA/D is comparable with C-NSGA-III, and both of them gain the best performance on 33% problems. As observed in Table S-9 in the supplement, however, CIC-MOEA/D shows a significant improvement over C-NSGA-III if we consider the number of both the best and the second-best HV results obtained. As expected, Fig. 6 (a) shows that CIC-MOEA/D, as well as PPS-MOEA/D, performs better on Type I′/II′ and Type II than on Type III. On the contrary, C-NSGA-III presents better performance on Type III than on the other two types.

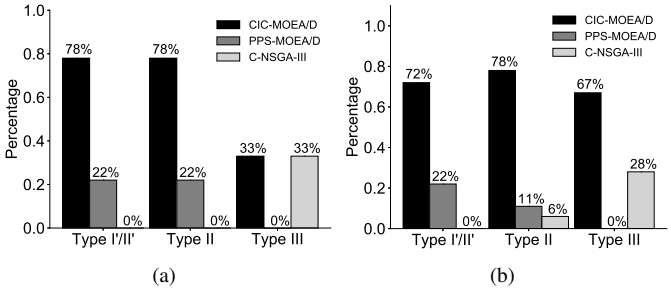


Fig. 6. The percentage of each type of DAS test problems on which the best HV results are obtained. (a): DAS-CMOPs; (b): DAS-CMaOPs.

TABLE III
SUMMARY OF COMPARISONS REGARDING HV ON DAS-CMAOPS.

CIC-MOEA/D v.s.	•	‡	○
PPS-MOEA/D	78%	9%	13%
C-NSGA-III	76%	7%	17%
C-MOEA/D	89%	2%	9%
NSGA-II-CDP	91%	2%	7%
C-MOEA/DD	94%	2%	4%

Similarly, we analyze the performance of the algorithms on DAS-CMaOPs. The statistical test results are summarized in Table III[6]. As seen, CIC-MOEA/D outperforms its competitors on at most 94% and at least 76% test problems. Clearly, the proposed CIC-MOEA/D is the best-performing algorithm on DAS-CMaOPs. Moreover, Fig. 6 (b) presents percentages of each type of problems on which the best HV results are obtained by CIC-MOEA/D, PPS-MOEA/D and C-NSGA-III. It can be found that CIC-MOEA/D, outperforming dramatically the other two algorithms, obtains the best HV results on 72% Type-I′/-II′, 78% Type-II and 67% Type-III problems. Similar to the behaviors on DAS-CMOPs, CIC-MOEA/D shows better performance on Type I′/II′ and Type II than on Type III.

D. Results on the constrained OSPS problem

In this section, we investigate the performance of the algorithms on a practical problem in search-based software engineering, i.e., the constrained OSPS problem proposed in [39]. This problem aims at selecting optimal software products from a software product line, which is usually represented by a feature model (FM). This selection should be (nearly) optimal with respect to a set of objectives and a set of constraints. Mathematically, this is a large-scale (with more than 1000 decision variables in general) and discrete (each variable takes either 0 or 1) multi-objective optimization problem with

[5]According to [4], some DAS-CMOPs with convergence-hardness are Type-I′ problems, i.e., DAS-CMOP1-6, while some are Type-II′ problems, i.e., DAS-CMOP7-9. Notice that a common feature is that the convergence is hampered by infeasible regions in both types. Therefore, we use Type I′/II′ to group them to emphasize this common feature.

[6]The HV results on DAS-CMaOPs are available in Table S-11 in the online supplement.

constraints [39]. Using objectives and constraints defined in [39], we can construct 2-, 3- and 4-objective OSPS instances for each FM. In this paper, we use 12 real-world FMs, also adopted in [39], to construct 36 OSPS instances. For more details on how to construct these instances, please refer to the original study [39].

TABLE IV
SUMMARY OF COMPARISONS REGARDING HV AND IGD+ ON THE CONSTRAINED OSPS PROBLEM.

CIC-MOEA/D v.s.	HV	IGD+
	•/ ‡ /○	•/ ‡ /○
PPS-MOEA/D	86%/11%/ 3%	83%/17%/ 0%
C-NSGA-III	81%/14%/ 6%	78%/14%/ 8%
C-MOEA/D	83%/14%/ 3%	83%/14%/ 3%
NSGA-II-CDP	81%/11%/ 8%	78%/11%/11%
C-MOEA/DD	83%/ 3%/14%	81%/ 8%/11%
SATVaEA	72%/ 3%/25%	64%/ 8%/28%

Notice that the SATVaEA[7] [58], a tailored algorithm for the OSPS problem, is also included in the performance comparisons. To make it fair, all the algorithms use the same single-point crossover, bit-flip mutation and satisfiability solver based repair operator [39] in the decision space. The HV and IGD+ results are presented in Tables S-12 and S-13 in the online supplement. As seen from these tables, CIC-MOEA/D performs best in most of the instances regarding both metrics. Specifically, the algorithm obtains the best HV and IGD+ results in 69% (25/36) and 61% (22/36) instances, respectively. Considering the pairwise comparisons between CIC-MOEA/D and each peer algorithm, as shown in Table IV, CIC-MOEA/D outperforms other algorithms in at least 72% and 64% instances concerning HV and IGD+, respectively. It can be inferred from Table IV that SATVaEA is the most competitive algorithm to CIC-MOEA/D. Fig. S-5 in the online supplement compares the distribution of solutions obtained in three 3-objective instances constructed based on toybox, uClinux and 2.6.28.6-icse11. As observed in that figure, the solutions of CIC-MOEA/D are distributed more widely than those of others. In particular, CIC-MOEA/D is more effective in finding better solutions on the boundary.

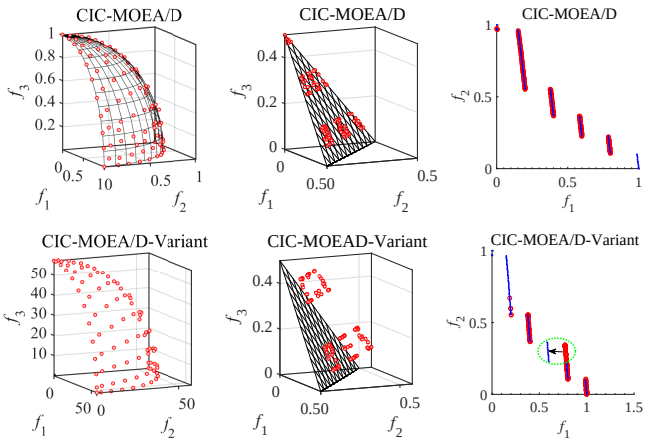
The above results indicate that the proposed CIC-MOEA/D gains the best overall performance on this real-world problem. According to [39], the constraints are mainly designed in such a way that the UPF is made partially feasible. In other words, most of the constructed OSPS instances are Type-II CMOPs. Therefore, the high performance of CIC-MOEA/D is not surprising because, as discussed previously, the mechanisms adopted in the algorithm fit well this type of problems.

VI. FURTHER INVESTIGATIONS

In this section, we start by showing the positive effect of ignoring constraints, and then investigate in depth why two evolutionary frameworks should be adopted in the proposed algorithm. In Section VI-C, CIC-MOEA/D is comprehensively compared with C-TAEA, a highly-related algorithm proposed recently.

A. On the effect of ignoring constraints

Experimental results in Section V have well validated the effectiveness of the proposed CIC-MOEA/D on particularly Type-I, -II, -I′ and -II′ problems. As discussed in Section I, ignoring constraints is one of the major reasons for the high performance of the proposed algorithm. In this section, we explicitly show that ignoring constraints can indeed make a difference by comparing CIC-MOEA/D with CIC-MOEA/D-Variant, a variant of the new algorithm which considers constraints throughout the evolutionary process. Table S-14 in the supplement gives medians of the HV results obtained by the aforementioned two algorithms. As shown clearly, CIC-MOEA/D performs better than, or at least comparably to, the variant in 31 out of 33 test instances (94%). In particular, CIC-MOEA/D significantly surpasses the variant in (almost) all the DC2-DTLZ1, DC2-DTLZ3, DC3-DTLZ1 and DC3-DTLZ3 test instances regardless of the number of objectives. As discussed in Section V-B, these problems are extremely difficult in terms of convergence because of local optima and infeasible barriers. By ignoring constraints, however, they can be well handled by our algorithm framework. As shown in Fig. 7, the solutions of CIC-MOEA/D converge (much) better than those of CIC-MOEA/D-Variant on three typical DC-DTLZ and DAS-CMOP test problems.



(a) DC2-DTLZ3 (I) (b) DC3-DTLZ1 (II′) (c) DAS-CMOP6 (I′)

Fig. 7. Final solutions obtained by CIC-MOEA/D and its variant on DC2-DTLZ3 (Type I), DC3-DTLZ1 (Type II′) and DAS-CMOP6 (Type I′).

B. Why two evolutionary frameworks?

Another prominent feature of CIC-MOEA/D is that it adopts two evolutionary frameworks in different stages. To be more specific, the first framework (F_1) used in the first stage performs a clustering-based ranking to the union of the parents and offsprings to select solutions for the next generation. In the second framework (F_2), a steady-state mechanism is employed to update sub-problems immediately

[7]The codes of SATVaEA can be found at https://www.researchgate.net/profile/Xiang_Yi9/publication/320007398_This_is_the_code_of_SATVaEA/data/59c71ff2458515548f32a02c/SATVaEArelease.zip

once a new solution has been generated. Then, is it really necessary to use two evolutionary frameworks in CIC-MOEA/D? In this section, we investigate in depth the benefits of using two frameworks by comparing CIC-MOEA/D with two variants, CIC-MOEA/D-F_1 and CIC-MOEA/D-F_2, where the used framework is unified to F_1 and F_2, respectively. Table S-15 in the supplement summarizes the HV results obtained by the three algorithms. As shown, CIC-MOEA/D performs better than the two variants in the majority of the test instances. More specifically, CIC-MOEA/D significantly outperforms CIC-MOEA/D-F_1 and CIC-MOEA/D-F_2 in 36 and 30 out of 44 test instances, respectively. Note that both CIC-MOEA/D-F_1 and CIC-MOEA/D-F_2 are able to obtain the best performance in some specific test instance, like for example, CIC-MOEA/D-F_1 on the 2-objective DC1-DTLZ1 and DC1-DTLZ3, and CIC-MOEA/D-F_2 on MW11 and MW12. Regarding the number of the best HV results, however, CIC-MOEA/D overwhelmingly outperforms the two variants. Therefore, the answer to the question posed at the beginning becomes clear. That is, it is indeed necessary to use two frameworks in the new proposal. In this way, by utilizing advantages of both frameworks, the performance of the algorithm can be promoted as much as possible.

C. Comparisons with C-TAEA

The C-TAEA[8], proposed by Li et al. [5], is one of the latest algorithms proposed for CMOPs. The algorithm uses two collaborative archives simultaneously. Note that constraint violations are not taken into consideration when updating the diversity-oriented archive. As both CIC-MOEA/D and C-TAEA maintain archives and both ignore constraints in a certain way, we make a special comparison between CIC-MOEA/D and C-TAEA in an independent section. Due to page limit, the HV results obtained by the two algorithms on four types of test problems are given in Table S-16 in the supplement. As seen, CIC-MOEA/D performs better than, equally to and worse than C-TAEA in 52%, 16% and 32% test instances, respectively. It is also found that C-TAEA significantly outperforms CIC-MOEA/D on Type-I test problems, but performs in general worse on Type-II, Type-II′ and Type-III problems. Considering the overall performance, however, CIC-MOEA/D is better than C-TAEA. The visualized comparisons of the solutions obtained by the two algorithms are presented in Fig. S-6 in the supplement.

Having compared the two algorithms with respect to the effectiveness, we further compare them in terms of the computational efficiency. Compared with CIC-MOEA/D, as shown in Fig. 8, C-TAEA runs much slower, requiring at least 11 times more runtime on the 3-objective DC-DTLZ problems. Note that the largest ratio of runtime is 24 (which is found on DC1-DTLZ3 and DC2-DTLZ1), indicating that the runtime of CIC-MOEA/D on the two problems is only $1/24$ (nearly 4%) of the time consumed by C-TAEA. The above results suggest that CIC-MOEA/D is (much) more computationally efficient than C-TAEA. The following are some possible explanations. First and foremost, C-TAEA maintains two archives simultaneously, one for convergence and one for diversity, whereas CIC-MOEA/D maintains only one archive. Second, the two archives in C-TAEA are updated at each generation. Different from this, the archive in CIC-MOEA/D is updated for every gap (e.g., 10) generations, and this reduces dramatically the time cost.

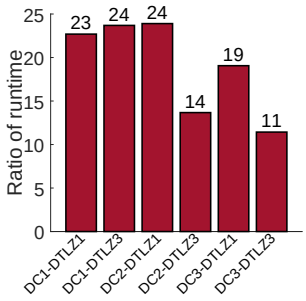


Fig. 8. Runtime comparisons between CIC-MOEA/D and C-TAEA on the 3-objective DC-DTLZ problems. The y-axis is the ratio of runtime, i.e., *runtime* (C-TAEA)/*runtime* (CIC-MOEA/D).

VII. Conclusions and Future Work

In this paper, we propose a simple yet effective decomposition-based algorithm for constrained multi-/many-objective optimization problems. In the algorithm, the evolutionary process is automatically divided into two stages, in which different constraint handling techniques and different evolutionary frameworks are adopted. In the first stage, constraints are entirely ignored, and the population is evolved by optimizing objectives only using a clustering-based ranking framework. In the second stage, constraints are emphasized over objectives to approximate well the constrained PF, and a steady-state framework as in the original MOEA/D is applied as usual.

The proposed algorithm is comprehensively compared with several state-of-the-art CMOEAs on a large number of artificial and real-world problems, which are all suggested recently. The experimental results well demonstrate the effectiveness and efficiency of the new proposal. The following are some main conclusions that we draw.

- The proposed algorithm obtains the best overall performance on the problems considered. Notably, the new algorithm performs particularly well on Type-I, Type-II, Type-I′ and Type-II′ problems. The common features of these problems are that the CPFs come either entirely or partly from the unconstrained counterparts, and/or that there are infeasible regions blocking the way of convergence. As discussed early in Section I, ignoring constraints can fit well these features, and this is indeed the truth according our experimental results. For Type-III problems, the proposed algorithm is also competitive to the state-of-the-art CMOEAs, e.g., C-NSGA-III.
- The proposed algorithm is computationally efficient. Concerning the actual running speed, the new proposal is faster than C-TAEA, C-MOEA/DD and C-NSGA-III. The

[8]The codes of C-TAEA are downloaded from the home page of Dr. K Li: https://cola-laboratory.github.io/docs/publications/

new algorithm is able to achieve a good tradeoff between effectiveness and efficiency.
- It is indeed necessary to use two frameworks in the new proposal. By utilizing advantages of both frameworks, the performance of the algorithm can be promoted as much as possible on a wide range of problems.

. Although the proposed algorithm obtains the best overall performance regarding the algorithms considered and the problems selected, it may be further improved on problems with some specific features. For example, in the future, we intend to enhance the algorithm on problems with narrow feasible regions [32] by using/desiging new constraint handling strategies, or on problems with diversity-hardness [4] by adapting weight vectors. Moreover, it is interesting to develop new implementations for the two frameworks to be used in the two stages. Finally, study on the applications of the algorithm to other real-world problems is also an important part of our future work.

REFERENCES

[1] H. Farzin, M. Fotuhifiruzabad, and M. Moeiniaghtaie, "A stochastic multi-objective framework for optimal scheduling of energy storage systems in microgrids," *IEEE Transactions on Smart Grid*, vol. 8, no. 1, pp. 117–127, 2017.

[2] C. Juang and Y. Yeh, "Multiobjective evolution of biped robot gaits using advanced continuous ant-colony optimized recurrent neural networks," *IEEE Transactions on Cybernetics*, vol. 48, no. 6, pp. 1910–1922, June 2018.

[3] G. Sun, H. Zhang, R. Wang, X. Lv, and Q. Li, "Multiobjective reliability-based optimization for crashworthy structures coupled with metal forming process," *Structural and Multidisciplinary Optimization*, vol. 56, no. 6, pp. 1571–1587, Dec 2017.

[4] Z. Fan, W. Li, X. Cai, H. Li, C. Wei, Q. Zhang, K. Deb, and E. Goodman, "Difficulty adjustable and scalable constrained multi-objective test problem toolkit," *Evolutionary Computation*, vol. 0, no. ja, pp. 1–28, 2019, pMID: 31120774. [Online]. Available: https://doi.org/10.1162/evco_a_00259

[5] 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, April 2019.

[6] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, "Constraint handling in multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 514–525, June 2009.

[7] C. A. Coello Coello, "Evolutionary multi-objective optimization: A historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, Nov. 2006.

[8] C. von Lücken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," *Computational Optimization and Applications*, vol. 58, no. 3, pp. 707–756, Jul 2014.

[9] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–35, Sep 2015.

[10] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 440–462, June 2017.

[11] S. Dutta and K. N. Das, "A survey on pareto-based eas to solve multi-objective optimization problems," in *Soft Computing for Problem Solving*, J. C. Bansal, K. N. Das, A. Nagar, K. Deep, and A. K. Ojha, Eds. Singapore: Springer Singapore, 2019, pp. 807–820.

[12] 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.

[13] S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization." *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 721–736, 2013.

[14] Y. Xiang, Y. Zhou, M. Li, and Z. Chen, "A vector angle based evolutionary algorithm for unconstrained many-objective problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 131–152, Feb. 2017.

[15] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[16] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[17] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[18] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, Oct 2015.

[19] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. 8th International Conference on Parallel Problem Solving from Nature, PPSN VIII*. Springer, 2004, pp. 832–842.

[20] J. Bader and E. Zitzler, "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, Jul. 2011.

[21] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[22] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11, pp. 1245 – 1287, 2002.

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

[24] T. Takahama and S. Sakai, "Constrained optimization by ϵ constrained particle swarm optimizer with ϵ-level control," in *Proc. of the 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology*. Muroran, Japan: Springer Berlin Heidelberg, May 2005, pp. 1019–1029.

[25] 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, Dec 2006.

[26] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, Feb 2008.

[27] 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, 2014.

[28] T. Ray, H. K. Singh, A. Isaacs, and W. Smith, *Infeasibility Driven Evolutionary Algorithm for Constrained Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 145–165.

[29] Y. Xiang, J. Peng, Y. Zhou, M. Li, and Z. Chen, "An angle based constrained many-objective evolutionary algorithm," *Applied Intelligence*, vol. 47, no. 3, pp. 705–720, Oct 2017.

[30] M. Asafuddoula, T. Ray, R. Sarker, and K. Alam, "An adaptive constraint handling approach embedded MOEA/D," in *2012 IEEE Congress on Evolutionary Computation*, June 2012, pp. 1–8.

[31] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. You, J. Mo, C. Wei, and E. Goodman, "An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions," *Soft Computing*, Feb 2019. [Online]. Available: https://doi.org/10.1007/s00500-019-03794-x

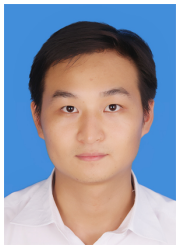
[32] Z. Fan, W. Li, X. Cai, H. Li, C. Wei, Q. Zhang, K. Deb, and E. Goodman, "Push and pull search for solving constrained multi-objective optimization problems," *Swarm and Evolutionary Computation*, vol. 44, pp. 665 – 679, 2019.

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

[34] W. Ying, W. He, Y. Huang, D. Li, and Y. Wu, "An adaptive stochastic ranking mechanism in MOEA/D for constrained multi-objective optimization," in *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, June 2016, pp. 514–518.

[35] Z. Fan, Yi Fang, W. Li, Jiewei Lu, X. Cai, and Caimin Wei, "A comparative study of constrained multi-objective evolutionary algorithms on constrained multi-objective optimization problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 209–216.

[36] Z. Liu, Y. Wang, and B. Wang, "Indicator-based constrained multiobjective evolutionary algorithms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2019.
[37] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.
[38] Z. Ma and Y. Wang, "Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2019.
[39] Y. Xiang, X. Yang, Y. Zhou, and H. Huang, "Enhancing decomposition-based algorithms by estimation of distribution for constrained optimal software product selection," *IEEE Transactions on Evolutionary Computation*, pp. 1–15, 2019.
[40] D. W. Coit, A. E. Smith, and D. M. Tate, "Adaptive penalty methods for genetic optimization of constrained combinatorial problems," *INFORMS Journal on Computing*, vol. 8, no. 2, pp. 173–182, 1996.
[41] A. Ben Hadj-Alouane and J. C. Bean, "A genetic algorithm for the multiple-choice integer program," *Operations research*, vol. 45, no. 1, pp. 92–101, 1997.
[42] H. Geng, M. Zhang, L. Huang, and X. Wang, "Infeasible elitists and stochastic ranking selection in constrained evolutionary multi-objective optimization," in *International Conference on Simulated Evolution and Learning*, T.-D. Wang, X. Li, S.-H. Chen, X. Wang, H. Abbass, H. Iba, G.-L. Chen, and X. Yao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 336–344.
[43] Z. Yang, X. Cai, and Z. Fan, "Epsilon constrained method for constrained multiobjective optimization problems: Some preliminary results," in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO Comp '14. New York, NY, USA: ACM, 2014, pp. 1181–1186.
[44] E. Mezura-Montes and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173 – 194, 2011.
[45] S. Zeng, R. Jiao, C. Li, X. Li, and J. S. Alkasassbeh, "A general framework of dynamic constrained multiobjective evolutionary algorithms for constrained optimization," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2678–2688, Sep. 2017.
[46] Y. Zhou, M. Zhu, J. Wang, Z. Zhang, Y. Xiang, and J. Zhang, "Tri-goal evolution framework for constrained many-objective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–14, 2018.
[47] 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, Oct 2019.
[48] B. Y. Qu and P. N. Suganthan, "Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods," *Engineering Optimization*, vol. 43, no. 4, pp. 403–416, 2011.
[49] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311 – 338, 2000.
[50] Y. Qi, Q. Zhang, X. Ma, Y. Quan, and Q. Miao, "Utopian point based decomposition for multi-objective optimization problems with complicated pareto fronts," *Applied Soft Computing*, vol. 61, pp. 844 – 859, 2017.
[51] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 264–283, April 2015.
[52] M. Li, S. Yang, and X. Liu, "Bi-goal evolution for many-objective optimization problems," *Artificial Intelligence*, vol. 228, pp. 45–65, 2015.
[53] Y. Xiang, Y. Zhou, X. Yang, and H. Huang, "A many-objective evolutionary algorithm with pareto-adaptive reference points," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2019.
[54] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable test problems for evolutionary multiobjective optimization*. Springer, 2005.
[55] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Difficulties in specifying reference points to calculate the inverted generational distance for many-objective optimization problems," in *IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*. IEEE, 2014, pp. 170–177.
[56] M. Li and X. Yao, "Quality evaluation of solution sets in multiobjective optimisation: A survey," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 26:1–26:38, Mar. 2019.
[57] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
[58] Y. Xiang, Y. Zhou, Z. Zheng, and M. Li, "Configuring Software Product Lines by Combining Many-Objective Optimization and SAT Solvers," *ACM Transactions on Software Engineering and Methodology*, vol. 26, no. 4, pp. 14:1–14:46, Feb. 2018.



Yi Xiang received the B.Sc. and M.Sc. degrees in mathematics from Guangzhou University, Guangzhou, China, in 2010 and 2013, respectively, and the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, in 2018. He is currently a Post-Doctoral Fellow working with Prof. X. W. Yang at the School of Software Engineering, South China University of Technology, Guangzhou. His current research interests include many-objective optimization and search-based software engineering.

Xiaowei Yang received the B.S. degree in theoretical and applied mechanics, the M.Sc. degree in computational mechanics, and the Ph.D. degree in solid mechanics from Jilin University, Changchun, China, in 1991, 1996, and 2000, respectively. He is currently a Full Time Professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. His current research interests include designs and analyses of algorithms for large-scale pattern recognitions, imbalanced learning, semi-supervised learning, support vector machines, tensor learning, and evolutionary computation. He has published over 100 journals and refereed international conference articles, including the areas of structural reanalysis, interval analysis, soft computing, support vector machines, and tensor learning.

Han Huang (M'15, SM'18) received the B. Man. degree in information management and information system from School of Mathematics, South China University of Technology, Guangzhou, China, in 2003, and the Ph.D. degree in computer science from the South China University of Technology, Guangzhou, China, in 2008. Currently, he is a Professor with the School of Software Engineering in SCUT. His research interests include theoretical foundation and application of evolutionary computation and stochastic heuristics. Dr. Huang is a senior member of CCF.

Jiahai Wang (M07, SM'19) received the Ph.D. degree from Toyama University, Toyama, Japan, in 2005. In 2005, he joined Sun Yat-Sen University, Guangzhou, China, where he is currently a professor with the School of Data and Computer Science. His main research interests include computational intelligence and its applications.