Push and Pull Search for Solving Constrained Multi-objective Optimization Problems

Zhun Fan[a,b,*], Wenji Li[a], Xinye Cai[c,d], Hui Li[e], Caimin Wei[f], Qingfu Zhang[g], Kalyanmoy Deb[h], Erik Goodman[h]

[a]Department of Electronic Engineering, Shantou University, Guangdong, China
[b]Key Lab of Digital Signal and Image Processing of Guangdong Province, Guangdong, China
[c]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Jiangsu, China
[d]Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China
[e]School of Mathematics and Statistics, Xi'an Jiaotong University, Shaanxi, China
[f]Department of Mathematics, Shantou University, Guangdong, China
[g]Department of Computer Science, City University of Hong Kong, Hong Kong, China
[h]BEACON Center for the Study of Evolution in Action, Michigan State University. East Lansing, Michigan, USA.

Abstract

This paper proposes a push and pull search (PPS) framework for solving constrained multi-objective optimization problems (CMOPs). To be more specific, the proposed PPS divides the search process into two different stages: push and pull search stages. In the push stage, a multi-objective evolutionary algorithm (MOEA) is used to explore the search space without considering any constraints, which can help to get across infeasible regions very quickly and to approach the unconstrained Pareto front. Furthermore, the landscape of CMOPs with constraints can be probed and estimated in the push stage, which can be utilized to conduct the parameter setting for the constraint-handling approaches to be applied in the pull stage. Then, a modified form of a constrained multi-objective evolutionary algorithm (CMOEA), with improved epsilon constraint-handling, is applied to pull the infeasible individuals achieved in the push stage to the feasible and non-dominated regions. To evaluate the performance regarding convergence and diversity, a set of benchmark CMOPs and a real-world optimization problem are used to test the proposed PPS (PPS-MOEA/D) and state-of-the-art CMOEAs, including MOEA/D-IEpsilon, MOEA/D-Epsilon, MOEA/D-CDP, MOEA/D-SR, C-MOEA/D and NSGA-II-CDP. The comprehensive experimental results show that the proposed PPS-MOEA/D achieves significantly better performance than the other six CMOEAs on most of the tested problems, which indicates the superiority of the proposed PPS method for solving CMOPs.

Keywords: Push and Pull Search, Constraint-handling Mechanisms, Constrained Multi-objective Evolutionary Algorithms, Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D)

1. Introduction

Many real-world optimization problems can be summarized as optimizing a number of conflicting objectives simultaneously with a set of equality and/or inequality constraints. Such problems are called constrained multi-objective optimization problems (CMOPs). Without lose of generality, a CMOP considered in this paper can be defined as follows [1]:

$$\begin{cases} \text{minimize} & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T \\ \text{subject to} & g_i(\mathbf{x}) \geq 0, i = 1, \ldots, q \\ & h_j(\mathbf{x}) = 0, j = 1, \ldots, p \\ & \mathbf{x} \in \mathbb{R}^n \end{cases} \tag{1}$$

where $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T$ is an m-dimensional objective vector, and $F(\mathbf{x}) \in \mathbb{R}^m$. $g_i(\mathbf{x}) \geq 0$ is an inequality constraint, and q is the number of inequality constraints. $h_j(\mathbf{x}) = 0$ is an equality constraint, and p represents the number of equality constraints. $\mathbf{x} \in \mathbb{R}^n$ is an n-dimensional decision vector.

When solving CMOPs with inequality and/or equality constraints, we usually convert the equality constraints into inequality constraints by introducing an extremely small positive number δ. The detailed transformation is given as follows:

$$h_j(\mathbf{x})' \equiv \delta - |h_j(\mathbf{x})| \geq 0 \tag{2}$$

To deal with a set of constraints in CMOPs, the overall constraint violation is a widely used approach, which summarizes the violations into a single scalar as follows:

$$\phi(\mathbf{x}) = \sum_{i=1}^{q} |\min(g_i(\mathbf{x}), 0)| + \sum_{j=1}^{p} |\min(h_j(\mathbf{x})', 0)| \tag{3}$$

Given a solution $\mathbf{x}^k \in \mathbb{R}^n$, if $\phi(\mathbf{x}^k) = 0$, $\mathbf{x}^k$ is feasible. All the feasible solutions constitute a feasible solution set S, which is defined as $S = \{\mathbf{x}|\phi(\mathbf{x}) = 0, \mathbf{x} \in \mathbb{R}^n\}$. For any two solutions $\mathbf{x}^a, \mathbf{x}^b \in S$, $\mathbf{x}^a$ is said to dominate $\mathbf{x}^b$ if $f_i(\mathbf{x}^a) \leq f_i(\mathbf{x}^b)$ for each $i \in \{1, ..., m\}$ and $f_j(\mathbf{x}^a) < f_j(\mathbf{x}^b)$ for at least one $j \in \{1, \ldots, m\}$, denoted as $\mathbf{x}^a \preceq \mathbf{x}^b$. If there is no other solution in S dominating solution $\mathbf{x}^*$, then $\mathbf{x}^*$ is called a Pareto optimal solution. All of the Pareto optimal solutions constitute a Pareto optimal set (PS). The mapping of the PS in the objective space is called a

*Corresponding author
Email address: zfan@stu.edu.cn (Zhun Fan)

Pareto optimal front (PF), which is defined as $PF = \{F(\mathbf{x})|\mathbf{x} \in PS\}$.

A key issue in CMOEAs is to maintain a balance between minimizing the objectives and satisfying the constraints. In fact, most constraint-handling mechanisms in evolutionary computation are designed to try to achieve this balance. For example, the penalty function approach adopts a penalty factor λ to maintain the balance between minimizing the objectives and satisfying the constraints. It converts a CMOP into an unconstrained MOP by adding the overall constraint violation multiplied by a predefined penalty factor λ to each objective [2]. In the case of $\lambda = \infty$, it is called a death penalty approach [3], which means that infeasible solutions are totally unacceptable. If λ is a static value during the search process, it is called a static penalty approach [4]. If λ is changing during the search process, it is called a dynamic penalty approach [5]. In the case in which λ is changing according to the information collected during the search process, it is called an adaptive penalty approach [6, 7, 8, 9].

In order to avoid the need to tune the penalty factors, another type of constraint-handling method is also in use, which compares the objectives and constraints separately. Representative examples include the constraint dominance principle (CDP) [10], epsilon constraint-handling method (EC) [11], stochastic ranking approach (SR) [12], and so on. In CDP [10], three basic rules are adopted to compare any two solutions. In the first rule, given two solutions $\mathbf{x}^i, \mathbf{x}^j \in \mathbb{R}^n$, if $\mathbf{x}^i$ is feasible and $\mathbf{x}^j$ is infeasible, $\mathbf{x}^i$ is better than $\mathbf{x}^j$. If $\mathbf{x}^i$ and $\mathbf{x}^j$ are both infeasible, the one with a smaller constraint violation is better. In the last rule, $\mathbf{x}^i$ and $\mathbf{x}^j$ are both feasible, and the one dominating the other is better. CDP is a popular constraint-handling method, as it is simple and has no extra parameters. However, it is not suitable for solving CMOPs with very small and narrow feasible regions [13]. For many generations, most or even all solutions in the working population are infeasible when solving CMOPs with this property. In addition, the diversity of the working population can hardly be well maintained, because the selection of solutions is only based on the constraint violations according to the second rule of CDP.

In order to solve CMOPs with small and narrow feasible regions, the epsilon constraint-handling (EC) [11] approach has been suggested. It is similar to CDP except for the relaxation of the constraints. In EC, the relaxation of the constraints is controlled by the epsilon level ε, which can help to maintain the diversity of the working population in the case when most solutions are infeasible. To be more specific, if the overall constraint violation of a solution is less than ε, this solution is deemed feasible. The epsilon level ε is a critical parameter in EC. In the case of $\varepsilon = 0$, EC is the same as CDP. Although EC can be used to solve CMOPs with small feasible regions, controlling the value of ε properly is not at all trivial.

Both CDP [10] and EC [11] first compare the constraints, then compare the objectives. SR [12] is different from CDP and EC in terms of the order of comparison. It adopts a probability parameter $p_f \in [0, 1]$ to decide if the comparison is to be based on objectives or constraints. For any two solutions, if a random number is less than p_f, the one with the non-dominated objectives is deemed better—i.e., the comparison is based on objectives. On the other hand, if the random number is greater than p_f, the comparison is based first on the constraints, then on the objectives, as is the case with CDP. In the case of $p_f = 0$, SR is equivalent to CDP.

In recent years, much work has been done in the field of many-objective evolutionary algorithms (MaOEAs) [14], which gives us new ways to solve CMOPs. In order to balance the constraints and the objectives, some researchers adopt multi-objective evolutionary algorithms (MOEAs) or MaOEAs (when the number of objectives is greater than three) to deal with constraints [15]. For an M-objective CMOP, its constraints can be converted into one or k extra objectives. Then the M-objective CMOP is transformed into an $(M + 1)$- or $(M + k)$-objective unconstrained MOP, which can be solved by MOEAs or MaOEAs. Representative examples include Cai and Wang's Method (CW) [16], the infeasibility driven evolutionary algorithms (IDEA) [17], and dynamic constrained multiobjective evolutionary algorithms [18].

To maintain a good balance between minimizing the objectives and satisfying the constraints, some researchers combine several constraint-handling mechanisms, which can be further divided into two categories, including adopting different constraint-handling mechanisms in either different evolutionary stages or in different subproblems. For example, the adaptive trade-off model (ATM) [19] uses two different constraint-handling mechanisms, including a multi-objective approach and adaptive penalty functions, in different evolutionary stages. The ensemble of constraint-handling methods (ECHM) [20] uses three different constraint-handling techniques, including epsilon constraint-handling (EC) [11], self-adaptive penalty functions (SP) [9] and superiority of feasible solutions (SF) [21]. Three subpopulations are generated in ECHM, and each subpopulation uses a different constraint-handling method.

In this paper, we propose a biphasic CMOEA, namely push and pull search (PPS), to balance objective minimization and constraint satisfaction. Unlike the above-mentioned constraint-handling methods, the PPS divides the search process into two different stages. In the first stage, only the objectives are optimized, which means the working population is pushed toward the unconstrained PF without considering any constraints. Furthermore, the landscape of constraints in CMOPs can be estimated in the push stage, which can be applied to conduct the parameter setting of the constraint-handling approaches to be applied in the pull stage. In the pull stage, an improved epsilon constraint-handling approach is adopted to pull the working population to the constrained PF. In summary, it provides a new framework and has the following potential advantages.

1. It has the ability to get across large infeasible regions of the constrained PF. Since the constraints are ignored in the push stage, any infeasible regions encountered before the true PF present no barriers for the working population.
2. It facilitates the parameter setting in the constraint-handling methods. Since the landscape of constraints has already been explored by the push process, much information has been discovered and gathered to guide the param-

eter setting for the pull stage.

The rest of the paper is organized as follows. Section 2 introduces the general idea of PPS. Section 3 gives an instantiation of PPS in the framework of MOEA/D, called PPS-MOEA/D. Section 4 designs a set of experiments to compare the proposed PPS-MOEA/D with six other CMOEAs, including MOEA/D-IEpsilon [22], MOEA/D-Epsilon [23], MOEA/D-SR [24], MOEA/D-CDP [24], C-MOEA/D [25] and NSGA-II-CDP [10]. Then, a real-world optimization problem, namely the robot gripper optimization, is used to test the performance of PPS-MOEA/D and the other six CMOEAs in Section 5. Finally, conclusions are drawn in section 6.

2. The General Framework of Push and Pull Search

Constraints define infeasible regions in the decision space, and sometimes are defined in such a way that they have an effect on the PF in the objective space. The influence of infeasible regions on PFs can be generally classified into three different situations. For each situation, the search behavior of PPS is illustrated by Fig. 1-3, respectively, which can be summarized as follows.

1. Infeasible regions block the way towards the PF, as illustrated by Fig. 1(a). In this circumstance, the unconstrained PF is the same as the constrained PF, and PPS has significant advantages compared with other CMOEAs. Since the constraints are ignored in the push stage of PPS, the infeasible regions have no effect on the searching of PPS. Fig. 1(a)-(e) show the push process at various stages, showing that the working population crosses the infeasible regions in this case without any extra effort. Because the constrained PF is the same as the unconstrained PF, the true PF has already been approximated by the working population in the push process, so the pull search has no effect on the working population, as shown in Fig. 1(f).
2. The unconstrained PF is covered by infeasible regions and all of it is infeasible. Every constrained Pareto optimal point thus lies on some constraint boundary, as illustrated by Fig. 2(a). In this circumstance, PPS first approaches the unconstrained PF by using the push strategy as illustrated by Fig. 2(a)-(c). After the working population approaches the unconstrained PF, the pull strategy is applied to pull the working population towards the true (constrained) PF, as illustrated by Fig. 2(d)-(f).
3. Infeasible regions make the original unconstrained PF partially feasible, as illustrated by Fig. 3(a). In this situation, some parts of the true PF have already been achieved during the push search, as illustrated by Fig. 3(c). In the pull stage, infeasible solutions are pulled to the feasible and non-dominated regions, as illustrated by Fig. 3(d)-(f). Finally, the entire true PF has been found by PPS. It is worth noting that infeasible regions may reduce the dimensionality of the PF in this situation.

From the above analysis, it can be observed that PPS can deal with CMOP situations with all types of interactions among constraints and the unconstrained PF.

The main steps of PPS includes the push and pull search processes. However, the decision as to when to switch from the push to the pull search process is very critical. A strategy for when to switch the search behavior is suggested as follows.

$$r_k \equiv \max\{rz_k, rn_k\} \leq \epsilon \tag{4}$$

where r_k represents the max rate of change between the ideal and nadir points during the last l generations. ϵ is a user-defined parameter; for the examples in this paper, we have set $\epsilon = 1e{-}3$. The rates of change of the ideal and nadir points during the last l generations are defined in Eq. (5) and Eq. (6), respectively.

$$rz_k = \max_{i=1,\ldots,m} \{\frac{|z_i^k - z_i^{k-l}|}{\max\{|z_i^{k-l}|, \Delta\}}\} \tag{5}$$

$$rn_k = \max_{i=1,\ldots,m} \{\frac{|n_i^k - n_i^{k-l}|}{\max\{|n_i^{k-l}|, \Delta\}}\} \tag{6}$$

where $z^k = (z_1^k, \ldots, z_m^k), n^k = (n_1^k, \ldots, n_m^k)$ are the ideal and nadir points in the k-th generation, and $z_i^k = \min_{j=1,\ldots,N} f_i(\mathbf{x}^j), n_i^k = \max_{j=1,\ldots,N} f_i(\mathbf{x}^j)$. N is the population size. $z^{k-l} = (z_1^{k-l}, \ldots, z_m^{k-l}), n^{k-l} = (n_1^{k-l}, \ldots, n_m^{k-l})$ are the ideal and nadir points in the $(k-l)$-th generation. rz_k and rn_k are two points in the interval $[0, 1]$. Δ is a very small positive number, which is used to make sure that the denominators in Eq. (5) and Eq. (4) are not equal to zero. In this paper, Δ is set to $1e-6$.

At the beginning of the search, r_k is initialized to 1.0. At each generation, r_k is updated according to Eq. (4). If r_k is less than or equal to the predefined threshold ϵ, the search behavior is switched to the pull search.

To summarize, PPS divides its search process into two different stages: push search and pull search. During the first stage, push search, which disregards the constraints, is adopted to approximate the unconstrained PF. Once Eq. (4) is satisfied, pull search is used to pull any infeasible solutions to the feasible and non-dominated regions—constraints are fully considered. PPS terminates when a predefined halting condition is met. In the following section, we will describe the instantiation of the push and pull strategy in a MOEA/D framework in detail.

3. An Instantiation of PPS in MOEA/D

This section describes the details of an instantiation of the push search method and the pull search method in the framework of a particular type of MOEA/D search, thus capturing the entire PPS method.

3.1. The push search

In the push search stage, an unconstrained MOEA/D is used to search for non-dominated solutions without considering any constraints. When solving a MOP by using MOEA/D, we decompose the MOP into a set of single optimization subproblems

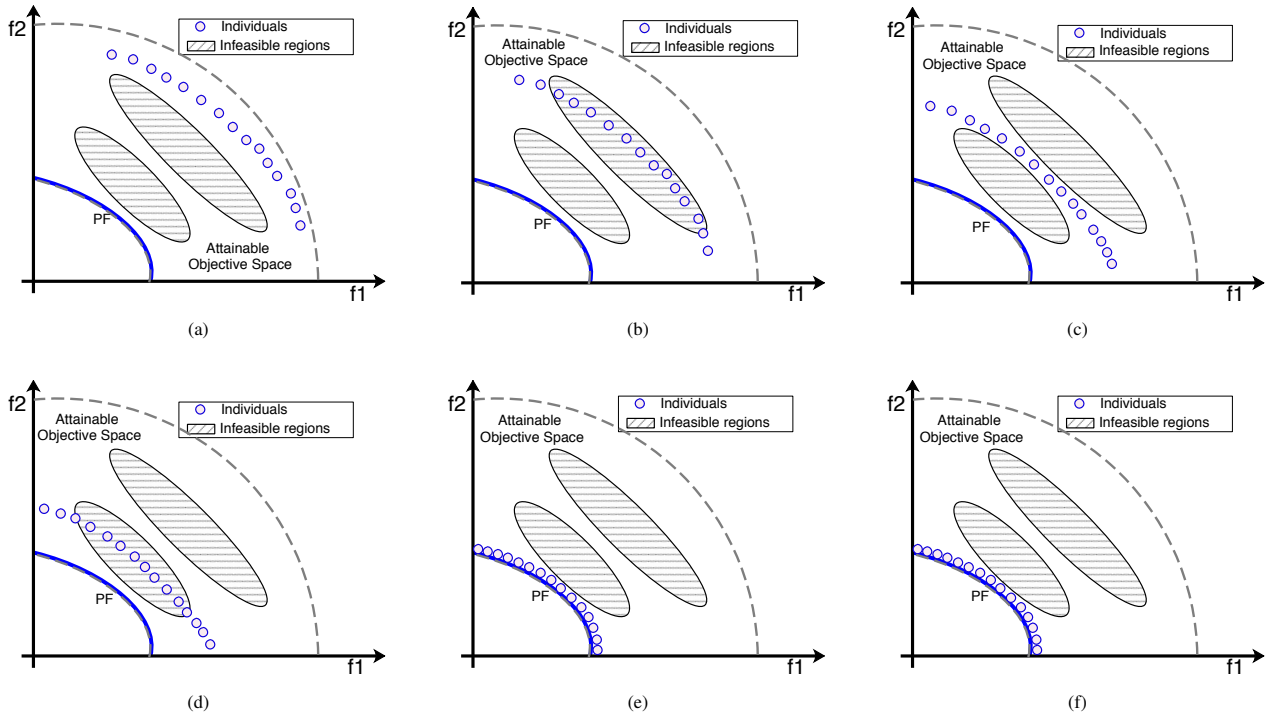


Figure 1: Infeasible regions block the way towards the PF, and the unconstrained PF is the same as the constrained PF. (a)-(e) show the various stages of the push search process, and show the working population getting across the infeasible regions without any extra efforts dealing with constraints. (f) shows the pull search process, which is the same as (e) in this particular case, since the true PF is the same as the unconstrained PF, and has already been achieved by the working population in the push search process.

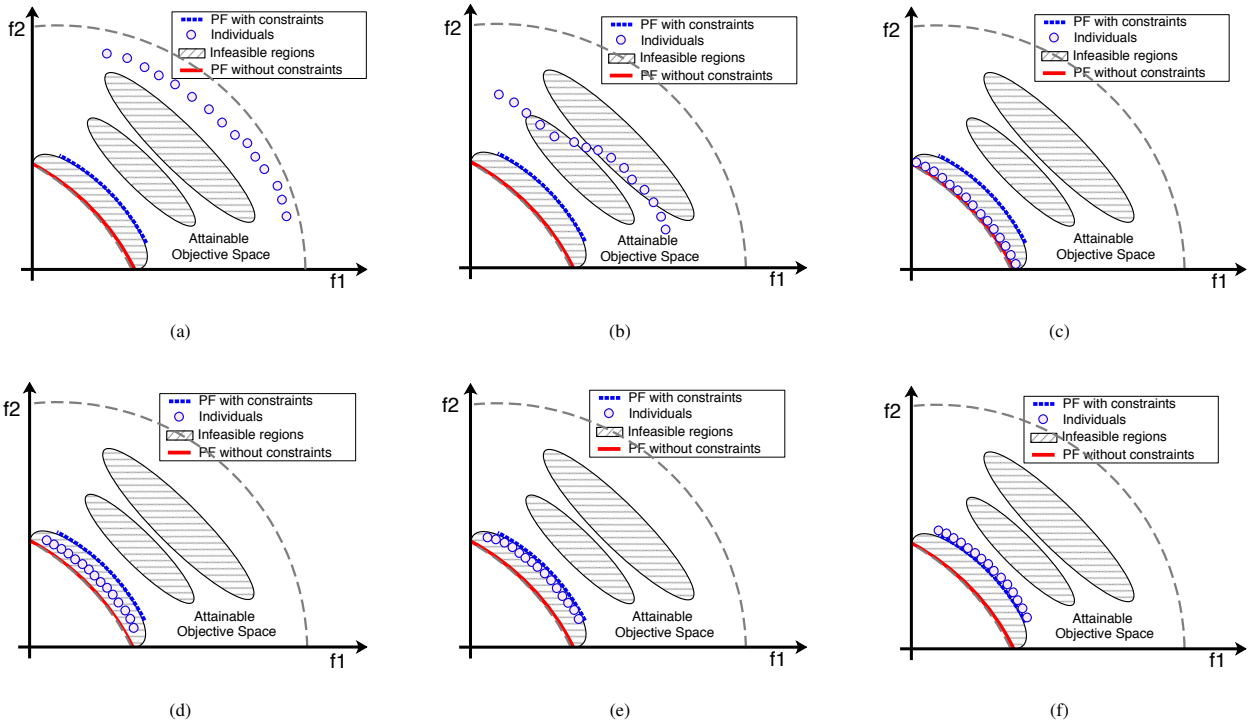


Figure 2: The unconstrained PF is covered by infeasible regions and all of it is infeasible. The true PF thus lies on some constraint boundaries. (a)-(c) show the push search process, in which the working population crosses the infeasible regions without any barriers. (d)-(f) show the pull search process, in which the infeasible solutions in the working population are gradually pulled to the feasible and non-dominated regions.

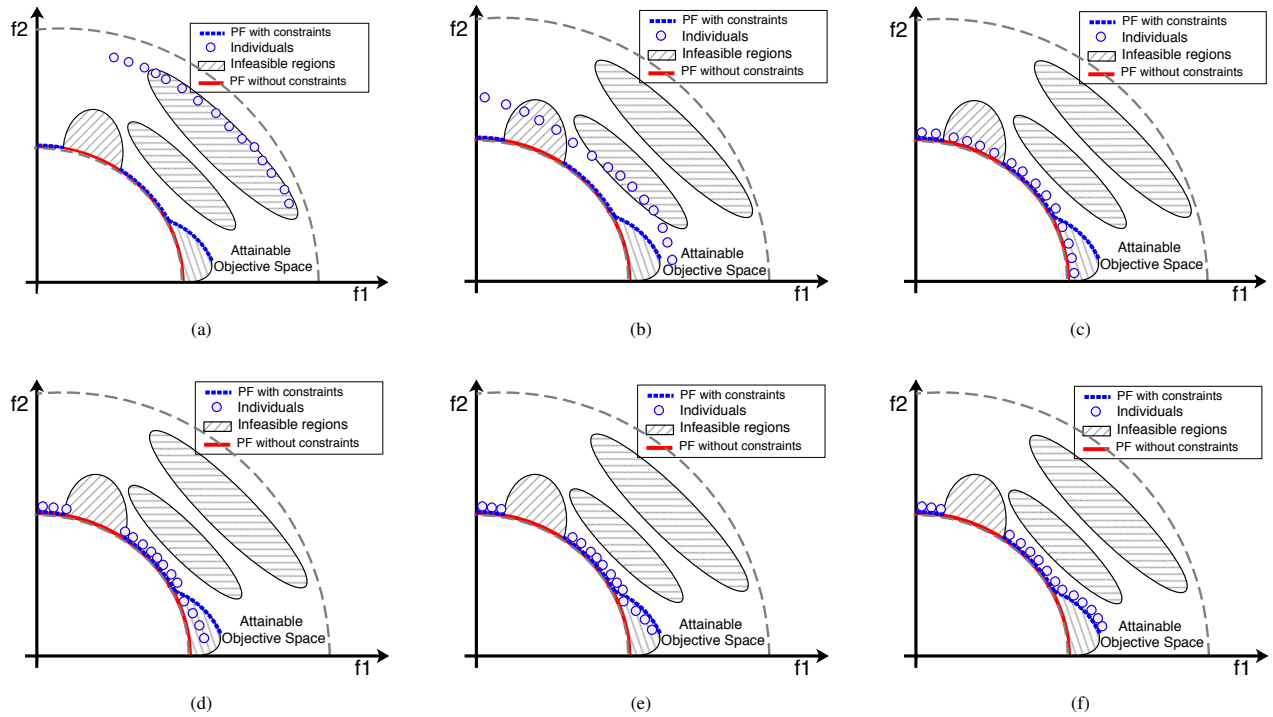


Figure 3: Infeasible regions make the original unconstrained PF partially feasible. (a)-(c) show the push search process, and some parts of the true PF have been found in this process. In the pull stage, infeasible solutions are gradually pulled to the feasible and non-dominated regions, as shown in (d)-(f).

and optimize them simultaneously in a collaborative way. Each subproblem is associated with a decomposition function by using a weight vector λ^i. In the decomposition-based selection approach, an individual is selected for survival into next generation based on the value of the decomposition function.

There are three popular decomposition approaches, including weighted sum [26], Tchebycheff [26] and boundary intersection approaches [27]. In this paper, we adopt the Tchebycheff decomposition method, with the detailed definition given as follows.

$$g^{te}(\mathbf{x}, \lambda^i, z^*) = \max_{j=1,\ldots,m} \frac{1}{\lambda_j^i}(|f_j(\mathbf{x}) - z_j^*|) \tag{7}$$

where λ^i is a weight vector, and $\sum_{j=1,\ldots,m} \lambda_j^i = 1, \lambda_j^i \geq 0$. z^* is the ideal point, and $z_j^* = \min_{k=1,\ldots,N} f_j(\mathbf{x}^k)$.

In the push search stage, a newly generated solution $\mathbf{x}$ is retained into the next generation based on the value of g^{te} as described in Algorithm 1.

Algorithm 1: Push Subproblem

```
Function result = PushSubproblems(x^j, y^i, z^*)
    result = false
    if g^te(y^i|λ^j, z^*) ≤ g^te(x^j|λ^j, z^*) then
        x^j = y^i
        result = true
    end
    return result
end
```

3.2. The pull search

In this process, infeasible solutions are pulled to the feasible and non-dominated regions. To achieve this, a constraint-handling mechanism is adopted to punish the infeasible solutions in the pull search stage. An improved epsilon constraint-handling to deal with constraints is proposed, with the detailed formulation given as follows.

$$\varepsilon(k) = \begin{cases} (1-\tau)\varepsilon(k-1), \text{if } rf_k < \alpha \\ \varepsilon(0)(1-\frac{k}{T_c})^{cp}, \text{if } rf_k \geq \alpha \end{cases} \tag{8}$$

where rf_k is the ratio of feasible to infeasible solutions in the k-th generation. τ is the parameter to control the speed of reducing the relaxation of constraints in the case of $rf_k < \alpha$, and $\tau \in [0, 1]$. α is to control the searching preference between the feasible and infeasible regions, and $\alpha \in [0, 1]$. cp is to control the speed of reducing relaxation of constraints in the case of $rf_k \geq \alpha$. $\varepsilon(k)$ is updated until the generation counter k reaches the control generation T_c. $\varepsilon(0)$ is set to the maximum overall constraint violation of the working population at the end of the push search. Compared with the ε setting in [28], the proposed method in Eq. (8) has an exponential decreasing speed to set $\varepsilon(k)$ in the case of $rf_k < \alpha$, which can help to find feasible solutions more quickly and efficiently. In the case of $rf_k \geq \alpha$, the Eq. (8) is the same as the ε setting in [28].

In the pull stage, a newly generated solution $\mathbf{x}$ is selected for survival into the next generation based on the value of g^{te}, the overall constraint violation $\phi(\mathbf{x})$ and the value of $\varepsilon(k)$, as illustrated by Algorithm 2. In the case of $\varepsilon(k) = 0$, it is the same as the constraint-handling method proposed for MOEA/D in [29].

Algorithm 2: Pull Subproblem

```
1  Function result = PullSubproblems(x^j,y^i,ε(k),z*)
2      result = false
3      if φ(y^i) ≤ ε(k) and φ(x^j) ≤ ε(k) then
4          if g^te(y^i|λ^j,z*) ≤ g^te(x^j|λ^j,z*) then
5              x^j = y^i; result = true
6          end
7      else if φ(y^i) == φ(x^j) then
8          if g^te(y^i|λ^j,z*) ≤ g^te(x^j|λ^j,z*) then
9              x^j = y^i; result = true
10         end
11     else if φ(y^i) < φ(x^j) then
12         x^j = y^i; result = true
13     end
14     return result
15 end
```

3.3. PPS Embedded in MOEA/D

Algorithm 3 outlines the pseudocode of PPS-MOEA/D. A CMOP is decomposed into N single-objective subproblems, and these subproblems are initialized at line 1. The ideal point z^* and the generation counter k are also initialized at line 1. At line 2, r_k, which is the maximum rate of change of ideal and nadir points, is initialized to 1.0, the flag of search stage is set to push (*PushStage* = *true*), and the maximum overall constraint violation found so far is initialized to -1 (*maxViolation* = -1). The ideal and nadir points at k-th generation are set at line 3, and the details can be found in Algorithm 4. *maxViolation* is updated at line 3, and the details can be found in Algorithm 5.

Then, the algorithm repeats lines 4-38 until T_{max} generations have been reached. The value of $\varepsilon(k)$ and the search strategy are set at lines 5-16 based on Eq. (4) and Eq. (8). At line 5, the maximum rate of change of ideal and nadir points r_k is calculated, and the details can be found in Algorithm 6. $\varepsilon(k)$ and $\varepsilon(0)$ are initialized at line 9. Lines 18-34 show the process of updating subproblems. At line 18, S represents the neighbor indexes of solution $\mathbf{x}^i$. Lines 19-20 perform a DE operator to generate a new solution $\overline{\mathbf{y}}$. Line 21 takes a polynomial mutation on $\overline{\mathbf{y}}$, and generates a new solution $\mathbf{y}^i$. Then the new solution $\mathbf{y}^i$ is repaired as follows: If an element of $\mathbf{y}^i$ is less than its lower boundary, it is reset to its lower boundary. If an element of $\mathbf{y}^i$ is great than its upper boundary, it is reset to its upper boundary. The ideal point $z*$ is updated at line 22, and the details can be found in Algorithm 7. At line 23, the maximum overall constraint violation—*maxViolation* is updated.

From line 27 to 31, it can be seen that different search strategies are used to update subproblems. When *PushStage* = *true*, the push search is adopted (line 28); otherwise, the pull search is used (line 30). At line 36, the generation counter k is updated. The ideal and nadir points at k-the generation are also set at this line, and the setting method can be found in Algorithm 4. Finally, the set of feasible and non-dominated solutions *NS* is updated according to the non-dominated ranking as given in NSGA-II [10] at line 37. The updating method can be found in Algorithm 8.

Algorithm 4 shows the process to set ideal and nadir points at k-th generation. At lines 2-5, the ideal and nadir points z^k, n^k are initialized. Lines 9-11 update each element of the ideal point z^k, and lines 12-14 update each element of the nadir point n^k.

Algorithm 5 shows the process to update the maximum overall constraint violation found so far. The updating process is performed at lines 2-4. If the newly generated solution $\mathbf{y}^i$ has a larger overall constraint violation ($\phi(\mathbf{y}^i)$) than *maxViolation*, *maxViolation* is set to $\phi(\mathbf{y}^i)$.

Algorithm 6 shows the pseudocode of calculating the maximum rate of change of ideal and nadir points r_k. At lines 2-3, rz_k and rn_k are calculated according to Eq. (5) and Eq. (6) respectively. At lines 4, r_k is calculated according to Eq. (4).

Algorithm 7 shows the pseudocode of updating the ideal point z^*. If the j-th objective of the newly generated solution $\mathbf{y}^i$ has a smaller value ($f_j(\mathbf{y}^i)$) than z_j^*, then z_j^* is set to $f_j(\mathbf{y}^i)$.

Algorithm 8 shows the pseudocode of selecting feasible and nondominated solutions. At lines 2-3, the number of feasible solutions (N_{fs}) and the set of feasible solutions ($\overline{P}_{fs}$) are calculated. If N_{fs} is smaller than the population size (N), the *result* is set to $\overline{P}_{fs}$, as shown at lines 4-5. Line 7 performs nondominated ranking on $\overline{P}_{fs}$, and solutions in $\overline{P}_{fs}$ are classified into q different fronts. Lines 9-12 add the first $k-1$ fronts to the *result* until the size of *result* is greater than ($N - \overline{P}_{fs}[k].size()$). Lines 13-16 select the $N - result.size()$ solutions front k-th front ($\overline{P}_{fs}[k]$) according to the crowding distance. The more details can be found in [10].

Remark: The proposed PPS is a general framework for solving CMOPs. Even though only PPS-MOEA/D is realized in this paper, it can be instantiated in many different MOEAs. At each search stage, a large variety of information can be gathered to extract useful knowledge that can be used to guide both search stages. In fact, knowledge discovery can be a critical step in the PPS framework. In this paper, we only utilize some statistical information. For example, the maximum overall constraint violation at the end of the push search stage is adopted to set the value of $\varepsilon(0)$. The ratio of feasible to infeasible solutions at the pull search stage is used to control the value of $\varepsilon(k)$. But in fact, many data mining methods and machine learning approaches can be integrated into the PPS framework for solving CMOPs more effectively and efficiently.

4. Experimental Study

4.1. Experimental Settings

To evaluate the performance of the proposed PPS method, six other CMOEAs, including MOEA/D-IEpsilon [22], MOEA/D-Epsilon [23], MOEA/D-SR [24], MOEA/D-CDP [24], C-MOEA/D [25] and NSGA-II-CDP [10], are tested on LIR-CMOP1-14 [22], which have large infeasible regions in the search space. The detailed parameters in each algorithm are listed as follows:

1. The mutation probability $Pm = 1/n$ (n denotes the dimension of a decision vector). The distribution index in the polynomial mutation is set to 20.

Algorithm 3: PPS-MOEA/D

Input: N: the number of subproblems. T_{max}: the maximum generation. N Weight vectors: $\lambda^1, \ldots, \lambda^N$. T: the size of the neighborhood. δ: the probability of selection from neighbors. n_r: the maximal number of solutions replaced by a child. T_c: the control generation for $\varepsilon(k)$.

Output: NS : a set of feasible non-dominated solutions

Decompose a CMOP into N subproblems associated with weight vectors. Generate a population $P = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ randomly. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i. Set the ideal point $z_j^* = \min_{i=1,\ldots,N} f_j(\mathbf{x}^i)$. Set $k = 1$.
Set r_k = 1.0, *PushStage* = *true*, *maxViolation* = -1;
SetIdealNadirPoints(P, k); For each $i = 1, \ldots, N$, *UpdateMaxViolation*($\mathbf{x}^i$,*maxViolation*);
while $k \leq T_{max}$ **do**
  if $k >= l$ **then** Set r_k = *CalcMaxChange*(k) ;
  if $k < T_c$ **then**
    if $r_k \leq \epsilon$ *and PushStage* == *true* **then**
      PushStage = *false*;
      $\varepsilon(k) = \varepsilon(0)$ = *maxViolation*;
    end
    if *PushStage* == $false$ **then**
      Update $\varepsilon(k)$ according to Eq. (8);
    end
  else
    $\varepsilon(k) = 0$;
  end
  for $i \leftarrow 1$ **to** N **do**
    if *rand* $< \delta$ **then** $S = B(i)$ **else** $S = \{1, \ldots, N\}$;
    Set $r^1 = i$ and select two indexes r^2 and r^3 from S randomly, and $r^2 \neq r^3$.
    Generate a new solution $\overline{\mathbf{y}}$ from x^{r^1}, x^{r^2} and x^{r^3} by a DE operator.
    Perform a polynomial mutation [30] on $\overline{\mathbf{y}}$ to generate a new solution $\mathbf{y}^i$, and repair $\mathbf{y}^i$.
  end
  UpdateIdealPoint$(\mathbf{y}^i, z^*)$;
  UpdateMaxViolation$(\mathbf{y}^i, maxViolation)$;
  Set $c = 0$.
  while $c \neq n_r$ *or* $S \neq \varnothing$ **do**
    select an element j from S randomly.
    if *PushStage* == *true* **then**
      result = *PushSubproblems*$(\mathbf{x}^j, \mathbf{y}^i, z^*)$;
    else
      result = *PullSubproblems*$(\mathbf{x}^j, \mathbf{y}^i, \varepsilon(k), z^*)$;
    end
    if result == true **then** $c = c + 1$;
    $S = S \setminus \{j\}$;
  end
  $k = k + 1$; *SetIdealNadirPoints*(P, k);
  NS = *NDSelect*$(NS \bigcup P, N)$;
end

Algorithm 4: Set Ideal and Nadir Points at k-th Generation

Function *SetIdealNadirPoints(P,k)*
  for $j \leftarrow 1$ **to** m **do**
    // m is the number of objective.
    $z_j^k = 1e30$; $n_j^k = -1e30$;
  end
  for $i \leftarrow 1$ **to** N **do**
    for $j \leftarrow 1$ **to** m **do**
      // $P(i)$ is the i-th solution in the population P
      if $f_j(P(i)) < z_j^k$ **then**
        $z_j^k = f_j(P(i))$
      end
      if $f_j(P(i)) > n_j^k$ **then**
        $n_j^k = f_j(P(i))$
      end
    end
  end
end

Algorithm 5: Update the Maximum Constraint Violation

Function *UpdateMaxViolation*($\mathbf{y}^i$,*maxViolation*)
  if $\phi(\mathbf{y}^i) > maxViolation$ **then**
    $maxViolation = \phi(\mathbf{y}^i)$ // according to Eq. (3);
  end
end

Algorithm 6: Calculate the Maximum Rate of Change of Ideal and Nadir Points r_k

Function r_k = *CalcMaxChange(k)*
  Calculate rz_k according to Eq. (5).
  Calculate rn_k according to Eq. (6).
  $r_k = \max\{rz_k, rn_k\}$, according to Eq. (4).
end

Algorithm 7: Update the Ideal Point

Function *UpdateIdealPoint*($\mathbf{y}^i, z^*$)
  for $j \leftarrow 1$ **to** m **do**
    // m is the number of objective.
    if $f_j(\mathbf{y}^i) < z_j^*$ **then**
      $z_j^* = f_j(\mathbf{y}^i)$
    end
  end
end

Algorithm 8: Select Feasible and Nondominated Solutions.

Function *result = NDSelect($\overline{P}, N$)*
  Calculate the number of feasible solutions N_{fs} in $\overline{P}$.
  Select a set of feasible solutions $\overline{P}_{fs}$ from $\overline{P}$.
  if $N_{fs} \leq N$ **then**
    $result = \overline{P}_{fs}$
  else
    Perform non-dominated ranking on $\overline{P}_{fs}$, and generate q fronts according to [10]. The set of solutions in the k-th front is denoted as $\overline{P}_{fs}[k]$.
    Set $k = 1$;
    while $(result.size() + \overline{P}_{fs}[k].size() < N)$ **do**
      $result = result \bigcup \overline{P}_{fs}[k]$;
      $k = k + 1$;
    end
    $remainNo = N - result.size()$;
    Calculate the crowding distance [10] for each solution in $\overline{P}_{fs}[k]$;
    Select top *remainNo* solutions with the maximum crowding distance from $\overline{P}_{fs}[k]$, denoted as $R_{solutions}$
    $result = result \bigcup R_{solutions}$;
  end
  return *result*
end

2. DE parameters: $CR = 1.0$, $f = 0.5$.
3. Population size: $N = 300$. Neighborhood size: $T = 30$.
4. Halting condition: each algorithm runs for 30 times independently, and stops when 300,000 function evaluations are reached.
5. Probability of selecting individuals from its neighborhood: $\delta = 0.9$.
6. The max number of solutions updated by a child: $nr = 2$.
7. Parameter setting in PPS-MOEA/D: $T_c = 800$, $\alpha = 0.95$, $\tau = 0.1$, $cp = 2$, $l = 20$.
8. Parameter setting in MOEA/D-IEpsilon: $T_c = 800$, $\alpha = 0.95$,$\tau = 0.1$, and $\theta = 0.05N$.
9. Parameter setting in MOEA/D-Epsilon: $T_c = 800$, $cp = 2$, and $\theta = 0.05N$.
10. Parameter setting in MOEA/D-SR: $S_r = 0.05$.

4.2. Performance Metric

To measure the performance of PPS-MOEA/D, MOEA/D-IEpsilon [22], MOEA/D-Epsilon [23], MOEA/D-SR [24], MOEA/D-CDP [24], C-MOEA/D [25] and NSGA-II-CDP [10], two popular metrics—the inverted generation distance (IGD) [31] and the hypervolume[32] are adopted.

- **Inverted Generational Distance** (IGD):

The IGD metric reflects the performance regarding convergence and diversity simultaneously. The detailed definition is given as follows:

$$\begin{cases} IGD(P^*, A) = \frac{\sum_{y^* \in P^*} d(y^*, A)}{|P^*|} \\ d(y^*, A) = \min_{y \in A} \{ \sqrt{\sum_{i=1}^{m} (y_i^* - y_i)^2} \} \end{cases} \tag{9}$$

where P^* denotes a set of representative solutions in the true PF, and A is an approximate PF achieved by a CMOEA. m denotes the number of objectives. For two-objective LIR-CMOPs with continuous PFs, 1000 points are sampled uniformly from the true PF to construct P^*. For LIR-CMOPs with three objectives, 10000 points are sampled uniformly from the true PF to constitute P^*. It is worth noting that a smaller value of IGD may indicate better performance with regards to diversity and/or convergence. (Note that this measure cannot be used if the true PF is unknown, so it is used primarily for benchmarking purposes.)

- **Hypervolume** (HV):

HV reflects the closeness of the set of non-dominated solutions achieved by a CMOEA to the true PF. The larger HV means that the corresponding non-dominated set is closer to the true PF.

$$HV(S) = VOL(\bigcup_{x \in S} [f_1(x), z_1^r] \times ... [f_m(x), z_m^r]) \tag{10}$$

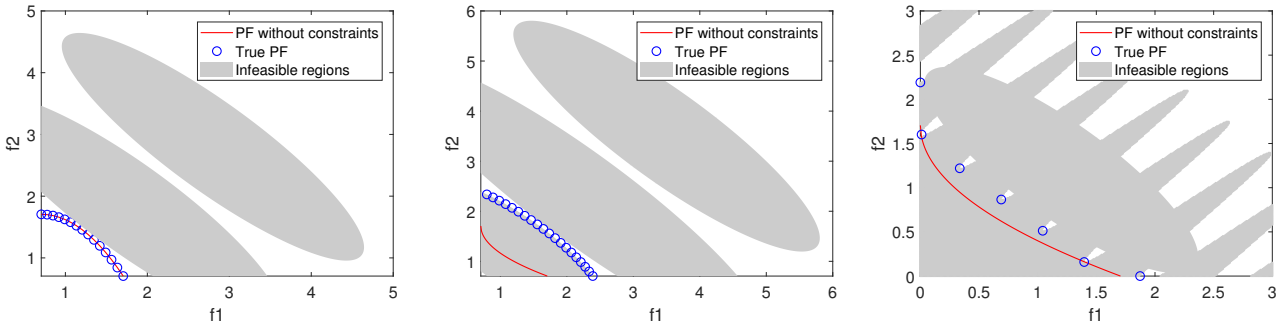
where $VOL(\cdot)$ is the Lebesgue measure, m denotes the number of objectives, $\mathbf{z}^r = (z_1^r, ..., z_m^r)^T$ is a user-defined reference point in the objective space. For each LIR-CMOP, the reference point is placed at 1.2 times the distance to the nadir point of the true PF (Note that this particular placement of the reference point also requires knowledge of the true PF). A larger value of HV may indicate better performance regarding diversity and/or convergence.

4.3. Discussion of Experiments

4.3.1. Comparisons among PPS-MOEA/D and the other six CMOEAs

The statistical results of the IGD values on LIR-CMOP1-14 achieved by PPS-MOEA/D and the other six CMOEAs in 30 independent runs are listed in Table 1. According to the Wilcoxon-Test in this table, it is clear that PPS-MOEA/D is significantly better than MOEA/D-Epsilon, MOEA/D-CDP, MOEA/D-SR, C-MOEA/D and NSGA-II-CDP on all of the fourteen tested problems in terms of the IGD metric. For LIR-CMOP1, LIR-CMOP4, LIR-CMOP9-10, and LIR-CMOP13, there are no statistically significant differences between PPS-MOEA/D and MOEA/D-IEpsilon. For the rest of nine test problems, PPS-MOEA/D is significantly better than MOEA/D-IEpsilon.

The statistical results of the HV values on LIR-CMOP1-14 achieved by PPS-MOEA/D and the other six CMOEAs in 30 independent runs are listed in Table 2. It can be observed that PPS-MOEA/D is significantly better than MOEA/D-Epsilon, MOEA/D-CDP, MOEA/D-SR, C-MOEA/D and NSGA-II-CDP on all the test instances in terms of the HV metric. For



(a) LIR-CMOP6 (b) LIR-CMOP7 (c) LIR-CMOP11

Figure 4: Illustrations of the feasible and infeasible regions of LIR-CMOP6, LIR-CMOP7 and LIR-CMOP11, corresponding to the three typical situations of infeasible regions influencing PFs as discussed in Section 2.

LIR-CMOP4, LIR-CMOP9-10, and LIR-CMOP13, there are no statistically significant differences between PPS-MOEA/D and MOEA/D-IEpsilon. For the rest of ten test instances, PPS-MOEA/D performs significantly better than MOEA/D-IEpsilon on these test problems. From the above observations, it is clear that the proposed PPS-MOEA/D achieves significantly better performance than the other six CMOEAs on most of the test problems.

To further discuss the advantages of the proposed PPS-MOEA/D, the populations achieved by each tested CMOEAs on LIR-CMOP6, LIR-CMOP7, and LIR-CMOP11 during the 30 independent runs with the median HV values are plotted in Fig. 5, Fig. 6 and Fig. 7, respectively. LIR-CMOP6, LIR-CMOP7 and LIR-CMOP11 are selected because they represent the three typical situations of infeasible regions influencing PFs as discussed in Fig. 1, Fig. 2 and Fig. 3, which are discussed in detail in the Section 2.

In particular, for LIR-CMOP6, there are two large infeasible regions in front of the PF, and the unconstrained PF is the same as the constrained PF, as illustrated by Fig. 4(a). In Fig. 5(a)-(b), we can observe that PPS-MOEA/D and MOEA/D-IEpsilon can get across the large infeasible regions, while the rest of CMOEAs, including MOEA/D-Epsilon, MOEA/D-CDP, MOEA/D-SR, C-MOEA/D and NSGA-II-CDP are trapped in the boundary of infeasible regions as shown in Fig. 5(c)-(g). The reason is constraints are ignored in PPS-MOEA/D at the push stage, and MOEA/D-IEpsilon uses the improved epsilon constraint-handling method to cross the infeasible regions (by dynamically adjusting the epsilon level to allow infeasible solutions to enter the population). As a result, the two large infeasible regions can not block the populations of PPS-MOEA/D and MOEA/D-IEpsilon to converge. However, the rest of five CMOEAs have no special mechanisms to cross large infeasible regions as illustrated in Fig. 4(a). As a result, the two large infeasible regions hinder their populations to converge. Therefore, they are trapped in the boundary of infeasible regions, which can be clearly observed in Fig. 5(c)-(g).

For LIR-CMOP7, there are three large infeasible regions, and the unconstrained PF is covered by infeasible regions and becomes no more feasible, as illustrated by Fig. 4(b). In Fig. 6(a)-(b), we can observe that PPS-MOEA/D and MOEA/D-IEpsilon can converge to the true PF, while the rest of five CMOEAs cannot converge to the true PF as shown in Fig. 6(c)-(g). The reason is that the three large infeasible regions can not hinder the populations of PPS-MOEA/D and MOEA/D-IEpsilon to converge. For PPS-MOEA/D, the unconstrained PF is first achieved at the push stage. Then, the population is pulled to the constrained PF by crossing only one infeasible region. Furthermore, the landscape of constraints has already been explored at the push process of PPS-MOEA/D. The maximum overall constraint violation can be calculated and applied to guide the ε parameter setting of the constraint-handling method in the pull stage of the PPS-MOEA/D properly, as defined in Eq. (8). Although MOEA/D-IEpsilon can get across the two large infeasible regions occasionally, in many cases, some individuals in MOEA/D-IEpsilon can not converge to the true PF, as shown in Fig. 6(b). The reason is that MOEA/D-IEpsilon has no mechanisms to explore the landscape of constraints in advance, thus lacking the potential of setting the ε parameter properly, as what can be done in PPS-MOEA/D. For the other five CMOEAs, they have no special mechanisms to cross the large infeasible regions as illustrated in Fig. 4(b). As a result, the two large infeasible regions in front of the constrained PF hinder their populations to converge. Therefore, the proposed PPS-MOEA/D can find the true PF of LIR-CMOP7 reliably, and MOEA/D-IEpsilon can find the true PF of LIR-CMOP7 occasionally. However, the rest of five CMOEAs are trapped in the boundary of infeasible regions, which can be clearly observed in Fig. 6(c)-(g).

For LIR-CMOP11, infeasible regions make the original unconstrained PF partially feasible, as illustrated by Fig. 4(c). The PF of LIR-CMOP11 is disconnected and has seven Pareto optimal solutions, among which two of them are located on the unconstrained PF, and five are not. PPS-MOEA/D can find all the Pareto optimal solutions as shown in Fig. 7(a), while the other six CMOEAs can not find all the seven Pareto optimal solutions, as shown in Fig. 7(b)-(g). The reason is that at the push stage the infeasible regions present no barriers for the population of PPS-MOEA/D, and the unconstrained PF of LIR-CMOP11 can be obtained at the push stage. Since two Pareto optimal solutions are situated at the unconstraint PF, PPS-MOEA/D can find these two Pareto optimal solutions at

the push stage instantly. Moreover, the landscape of constraints has already been explored at the push stage, which can help the searching of PPS-MOEA/D at the pull stage. The population of PPS-MOEA/D only needs to get cross one single infeasible region to find the other five Pareto optimal solutions at the pull stage. However, for the other six CMOEAs, they need to get cross several infeasible and overlapping regions to find the Pareto optimal solutions. Besides, the landscape of constraints are not well explored in advance during the search, which makes it difficult for the other six CMOEAs to set constraint-handling parameters properly. As a result, the proposed PPS-MOEA/D can find all the seven discrete Pareto optimal solutions reliably, while the other six CMOEAs can only find some Pareto optimal solutions occasionally.

According to the above observations and analysis, we can conclude that the proposed PPS-MOEA/D performs significantly better than the other six CMOEAs on most test cases. The experimental results demonstrate that the proposed PPS-MOEA/D can solve CMOPs well by taking advantage of both the push and pull strategies.

4.3.2. A Comparison between PPS-MOEA/D and its Variant

By replacing the Eq.(8) with that of Takahama's article [28] for PPS-MOEA/D, another version of PPS-MOEA/D is obtained, denoted as PPS-MOEA/D1. The same experimental parameters are adopted to test the performance of PPS-MOEA/D1.

The statistical results of the IGD values on LIR-CMOP1-14 achieved by PPS-MOEA/D1 and PPS-MOEA/D in 30 independent runs are shown in Table 3. According to the Wilcoxon-Test in this table, it is clear that PPS-MOEA/D is significantly better than PPS-MOEA/D1 on LIR-CMOP1-4 and LIR-CMOP7-8. PPS-MOEA/D1 is significantly better than PPS-MOEA/D on LIR-CMOP5-6, LIR-CMOP9-10, and LIR-CMOP13. For LIR-CMOP11-12 and LIR-CMOP14, there are not significant differences between PPS-MOEA/D and PPS-MOEA/D1.

The statistical results of the HV values on LIR-CMOP1-14 achieved by PPS-MOEA/D1 and PPS-MOEA/D in 30 independent runs are listed in Table 4. It can be observed that PPS-MOEA/D is significantly better than PPS-MOEA/D1 on LIR-CMOP1-4, LIR-CMOP8, LIR-CMOP13-14, and it is significantly worse than PPS-MOEA/D1 on LIR-CMOP5-6, LIR-CMOP9-10 and LIR-CMOP12. For the rest of test instances, there are not significant differences between PPS-MOEA/D and PPS-MOEA/D1.

For LIR-CMOP1-4, PPS-MOEA/D is significantly better than PPS-MOEA/D1 in terms of both IGD and HV metrics. A common feature of these problems is that they have narrow feasible regions, and their constrained PFs are far away from their unconstrained PFs. At the push stage, PPS-MOEA/D and PPS-MOEA/D1 both can find the unconstrained PFs. At the pull stage, since all the solutions are infeasible, PPS-MOEA/D adopts the first rule in Eq. (8) to decrease the epsilon value dramatically. Thus, PPS-MOEA/D can quickly get across infeasible regions and find feasible solutions efficiently. In contrast, PPS-MOEA/D1 decreases the epsilon value much more slowly as compared with PPS-MOEA/D, which may slow down the process to get across infeasible regions to a large extent. When PPS-MOEA/D find enough feasible solutions, it adopts the second rule in Eq. (8) to decrease the epsilon value slowly, which leads to a more thorough searching for feasible solutions. Therefore, PPS-MOEA/D performs significantly better than PPS-MOEA/D1 on LIR-CMOP1-4.

For LIR-CMOP5-6 and LIR-CMOP9-10, PPS-MOEA/D1 performs significantly better than PPS-MOEA/D. A common feature of LIR-CMOP5-6 and LIR-CMOP9-10 is that their constrained PFs are the same as their unconstrained counterparts, and located right on top of their unconstrained PFs. At the push stage, PPS-MOEA/D and PPS-MOEA/D1 can both find the unconstrained PFs. At the pull stage, PPS-MOEA/D1 decreases the epsilon value much more slowly as compared with PPS-MOEA/D, which lead to a more thorough searching for feasible solutions by using PPS-MOEA/D1. Therefore, for LIR-CMOP5-6 and LIR-CMOP9-10, PPS-MOEA/D1 performs significantly better than PPS-MOEA/D.

From the above observations and analysis, it is clear that the proposed PPS-MOEA/D performs better than PPS-MOEA/D1 in more cases. PPS-MOEA/D is more suitable for solving CMOPs with constrained PFs located far away from their unconstrained PFs as illustrated in Fig. 2(a), while PPS-MOEA/D1 is more suitable for solving CMOPs whose constrained PFs are located right on top of their unconstrained PFs as illustrated in Fig. 1(a).

4.3.3. The weakness of PPS-MOEA/D

As illustrated in Section 3, PPS-MOEA/D can be classified into two different search stages, including the push search and pull search. If the population of PPS-MOEA/D is converged to the unconstrained PF with only one point, it is difficult to pull the population to the constrained PF. Base on the above hypothesis, we design a CMOP named TNK-v1 which is based on TNK [33], whose unconstrained PF only has one point. The detail definition of TNK-v1 is given as follows:

$$\begin{cases} f_1(x) = x_1 + g_1(x) \\ f_2(x) = x_2 + g_2(x) \\ g_1(x) = \sum_{j\in J_1} (x_j - sin(0.5x_2))^2 \\ g_2(x) = \sum_{j\in J_2} (x_j - cos(0.5x_1))^2 \\ J_1 = \{3, 5, \ldots, 29\}, J_2 = \{4, 6, \ldots, 30\} \\ c_1(x) = x_1^2 + x_2^2 - 1.0 - 0.1\cos(16.0\arctan\frac{x_1}{x_2}) \geq 0 \\ c_2(x) = 0.5 - (x_1 - 0.5)^2 - (x_2 - 0.5)^2 \geq 0 \\ x_1, x_2 \in [1e-4, \pi] \text{ and } x_3, x_4, \ldots, x_{30} \in [0, 1] \end{cases} \tag{11}$$

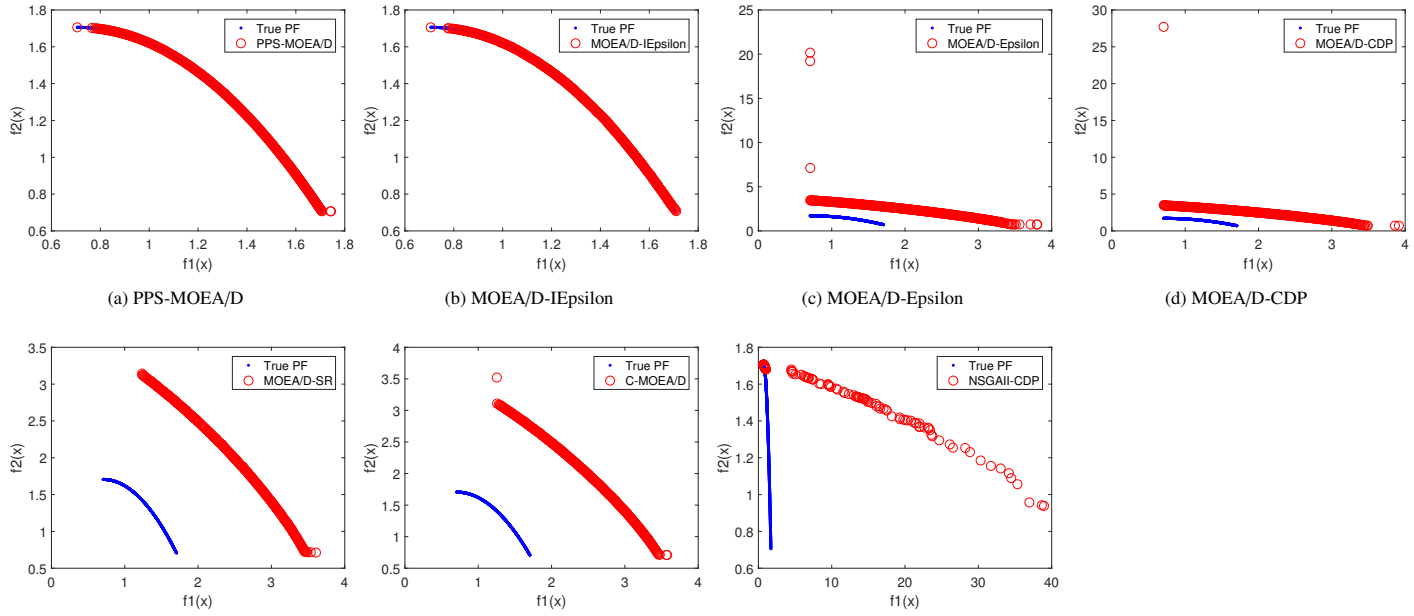
We use the same experimental parameters in Section 4.1 to test the seven CMOEAs on TNK-v1. The statistical results of the IGD and HV values on TNK-v1 achieved by each CMOEA are shown in Table 5. From this table, we can observe that PPS-MOEA/D is significantly worse than the other six CMOEAs on TNK-v1. In Fig. 8, we can observe that PPS-MOEA/D only find a part of the true PF. One possible reason is that, at the end of the push stage, the population of PPS-MOEA/D is converged to a single unconstrained Pareto optimal point. The diversity of

Table 1: IGD results of PPS and the other six CMOEAs on LIR-CMOP1-14. To facilitate the display of this table, PPS, IEpsilon, Epsilon, CDP, and SR are short for MOEA/D-PPS, MOEA/D-IEpsilon, MOEA/D-Epsilon, MOEA/D-CDP, and MOEA/D-SR respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between PPS-MOEA/D and each of the other six CMOEAs. † and ‡ denote that the performance of the corresponding algorithm is significantly worse than or better than that of PPS-MOEA/D, respectively. 'S-D-I' indicates PPS-MOEA/D is superior to, not significantly different from or inferior to the corresponding compared CMOEAs.

Test Instance		PPS	IEpsilon	Epsilon	CDP	SR	C-MOEA/D	NSGA-II-CDP
LIRCMOP1	mean	**6.41E-03**	7.97E-03	5.74E-02 †	1.11E-01†	1.81E-02†	1.26E-01†	3.23E-01†
	std	1.94E-03	3.55E-03	2.89E-02	5.04E-02	1.66E-02	7.03E-02	7.33E-02
LIRCMOP2	mean	**4.67E-03**	5.23E-03†	5.39E-02†	1.43E-01†	9.63E-03†	1.40E-01†	3.03E-01†
	std	7.84E-04	1.01E-03	2.13E-02	5.55E-02	7.23E-03	5.44E-02	7.24E-02
LIRCMOP3	mean	**8.55E-03**	1.13E-02†	8.81E-02†	2.61E-01†	1.78E-01†	2.80E-01†	4.08E-01†
	std	5.18E-03	6.42E-03	4.36E-02	4.33E-02	7.20E-02	4.21E-02	1.15E-01
LIRCMOP4	mean	**4.68E-03**	4.85E-03	6.51E-02†	2.53E-01†	1.95E-01†	2.59E-01†	3.85E-01†
	std	1.12E-03	2.05E-03	3.01E-02	4.34E-02	6.40E-02	3.51E-02	1.35E-01
LIRCMOP5	mean	**1.84E-03**	2.13E-03†	1.15E+00†	1.05E+00†	1.04E+00†	1.10E+00†	5.53E-01†
	std	9.26E-05	3.79E-04	1.98E-01	3.63E-01	3.66E-01	2.99E-01	6.88E-01
LIRCMOP6	mean	**2.49E-03**	2.33E-01†	1.27E+00†	1.09E+00†	9.43E-01†	1.31E+00†	5.74E-01†
	std	3.40E-04	5.06E-01	2.95E-01	5.20E-01	5.90E-01	2.08E-01	4.21E-01
LIRCMOP7	mean	**2.80E-03**	3.73E-02†	1.51E+00†	1.46E+00†	1.08E+00†	1.56E+00†	2.38E-01†
	std	9.85E-05	5.41E-02	5.09E-01	5.58E-01	7.58E-01	4.24E-01	4.06E-01
LIRCMOP8	mean	**2.78E-03**	2.75E-02†	1.62E+00†	1.38E+00†	1.01E+00†	1.58E+00†	6.02E-01†
	std	7.56E-05	5.92E-02	3.05E-01	6.15E-01	7.24E-01	3.71E-01	7.39E-01
LIRCMOP9	mean	9.94E-02	**4.98E-03**	4.90E-01†	4.81E-01†	4.85E-01†	4.81E-01†	6.44E-01†
	std	1.52E-01	1.37E-02	4.22E-02	5.24E-02	4.78E-02	5.24E-02	1.60E-02
LIRCMOP10	mean	**2.11E-03**	2.11E-03	2.13E-01†	2.16E-01†	1.92E-01†	2.13E-01†	5.97E-01†
	std	7.75E-05	7.11E-05	5.32E-02	6.81E-02	6.81E-02	4.63E-02	3.20E-02
LIRCMOP11	mean	**2.83E-03**	5.81E-02†	3.47E-01†	3.42E-01†	3.16E-01†	3.81E-01†	4.87E-01†
	std	1.36E-03	5.79E-02	9.28E-02	9.22E-02	7.49E-02	8.95E-02	1.05E-02
LIRCMOP12	mean	**2.70E-02**	3.36E-02†	2.52E-01†	2.69E-01†	2.06E-01†	2.50E-01†	5.80E-01†
	std	5.00E-02	5.18E-02	8.98E-02	9.06E-02	5.61E-02	9.63E-02	1.17E-01
LIRCMOP13	mean	**6.46E-02**	6.46E-02	1.20E+00†	1.21E+00†	8.86E-01†	1.18E+00†	1.39E+01†
	std	2.18E-03	1.64E-03	3.06E-01	3.17E-01	5.76E-01	3.78E-01	2.26E+00
LIRCMOP14	mean	**6.42E-02**	6.54E-02†	1.02E+00†	1.11E+00†	1.03E+00†	1.25E+00†	1.36E+01†
	std	1.69E-03	2.04E-03	4.86E-01	3.98E-01	4.70E-01	5.30E-02	2.17E+00
Wilcoxon-Test (S-D-I)		–	9-5-0	14-0-0	14-0-0	14-0-0	14-0-0	14-0-0

Table 2: HV results of PPS-MOEA/D and the other six CMOEAs on LIR-CMOP1-14. To facilitate the display of this table, PPS, IEpsilon, Epsilon, CDP, and SR in this table are short for MOEA/D-PPS, MOEA/D-IEpsilon, MOEA/D-Epsilon, MOEA/D-CDP, and MOEA/D-SR respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between PPS-MOEA/D and each of the other six CMOEAs. † and ‡ denote that the performance of the corresponding algorithm is significantly worse than or better than that of PPS-MOEA/D, respectively. 'S-D-I' indicates PPS-MOEA/D is superior to, not significantly different from or inferior to the corresponding compared CMOEAs.

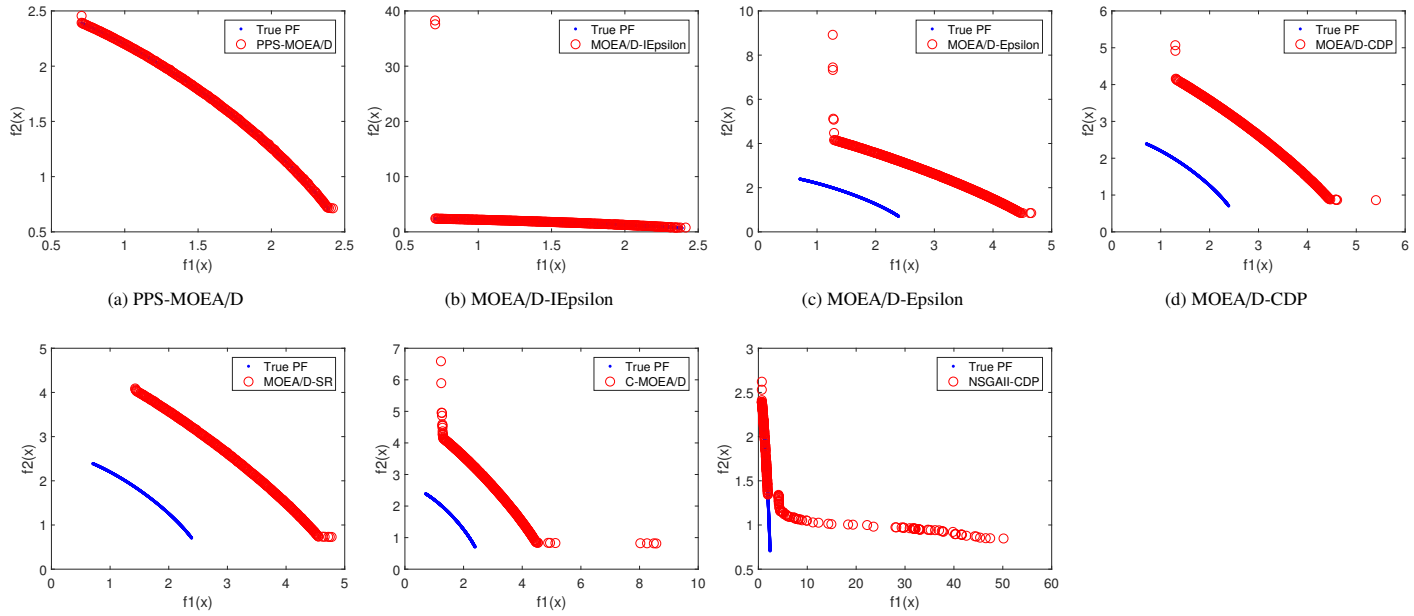
Test Instance		PPS	IEpsilon	Epsilon	CDP	SR	C-MOEA/D	NSGA-II-CDP
LIRCMOP1	mean	**1.02E+00**	1.01E+00†	9.59E-01†	7.54E-01†	9.96E-01†	7.41E-01†	5.16E-01†
	std	1.58E-03	2.43E-03	3.28E-02	8.95E-02	2.91E-02	1.22E-01	5.57E-02
LIRCMOP2	mean	**1.35E+00**	1.35E+00†	1.28E+00†	1.06E+00†	1.34E+00†	1.07E+00†	8.24E-01†
	std	1.01E-03	1.32E-03	2.88E-02	1.08E-01	1.47E-02	9.10E-02	1.15E-01
LIRCMOP3	mean	**8.70E-01**	8.68E-01†	7.98E-01†	4.86E-01†	5.91E-01†	4.71E-01†	4.08E-01†
	std	2.65E-03	3.92E-03	3.93E-02	4.31E-02	1.07E-01	4.09E-02	2.88E-02
LIRCMOP4	mean	**1.09E+00**	1.09E+00	1.02E+00	7.35E-01†	8.15E-01†	7.31E-01†	6.17E-01†
	std	2.47E-03	2.46E-03	4.19E-02	5.44E-02	8.70E-02	5.16E-02	1.06E-01
LIRCMOP5	mean	**1.46E+00**	1.46E+00†	4.30E-02†	1.63E-01†	1.82E-01†	9.72E-02†	9.39E-01†
	std	2.92E-04	1.33E-03	2.35E-01	4.43E-01	4.39E-01	3.70E-01	3.21E-01
LIRCMOP6	mean	**1.13E+00**	9.26E-01†	5.40E-02†	1.88E-01†	3.02E-01†	2.33E-02†	4.13E-01†
	std	1.77E-04	4.23E-01	2.21E-01	3.87E-01	4.62E-01	1.28E-01	1.89E-01
LIRCMOP7	mean	**3.02E+00**	2.86E+00†	3.03E-01†	3.74E-01†	9.88E-01†	2.04E-01†	2.40E+00†
	std	2.66E-03	1.96E-01	9.07E-01	9.58E-01	1.27E+00	7.52E-01	6.52E-01
LIRCMOP8	mean	**3.02E+00**	2.94E+00†	1.06E-01†	5.17E-01†	1.10E+00†	1.66E-01†	1.90E+00†
	std	1.14E-03	1.86E-01	5.49E-01	1.05E+00	1.20E+00	6.11E-01	7.56E-01
LIRCMOP9	mean	3.57E+00	**3.71E+00**	2.74E+00†	2.77E+00†	2.75E+00†	2.77E+00†	2.06E+00†
	std	2.24E-01	1.88E-02	1.48E-01	1.84E-01	1.64E-01	1.84E-01	1.08E-02
LIRCMOP10	mean	**3.24E+00**	3.24E+00	2.89E+00†	2.88E+00†	2.93E+00†	2.89E+00†	2.04E+00†
	std	3.08E-04	2.48E-04	1.02E-01	1.36E-01	1.35E-01	9.77E-02	4.45E-02
LIRCMOP11	mean	**4.39E+00**	4.23E+00†	3.34E+00†	3.35E+00†	3.38E+00†	3.24E+00†	3.11E+00†
	std	2.22E-04	1.84E-01	2.57E-01	2.57E-01	2.90E-01	2.55E-01	1.54E-02
LIRCMOP12	mean	**5.61E+00**	5.59E+00†	4.88E+00†	4.83E+00†	5.03E+00†	4.89E+00†	3.28E+00†
	std	1.53E-01	1.58E-01	3.17E-01	3.28E-01	1.75E-01	3.45E-01	3.61E-01
LIRCMOP13	mean	**5.71E+00**	5.71E+00	4.55E-01†	4.63E-01†	1.89E+00†	6.29E-01†	0.00E+00†
	std	1.27E-02	1.30E-02	1.30E+00	1.42E+00	2.57E+00	1.71E+00	0.00E+00
LIRCMOP14	mean	**6.19E+00**	6.18E+00†	1.33E+00†	8.81E-01†	1.27E+00†	1.80E-01†	0.00E+00†
	std	1.31E-02	1.09E-02	2.45E+00	1.97E+00	2.29E+00	2.60E-01	0.00E+00
Wilcoxon-Test (S-D-I)		–	10-4-0	14-0-0	14-0-0	14-0-0	14-0-0	14-0-0



(a) PPS-MOEA/D (b) MOEA/D-IEpsilon (c) MOEA/D-Epsilon (d) MOEA/D-CDP

(e) MOEA/D-SR (f) C-MOEA/D (g) NSGA-II-CDP

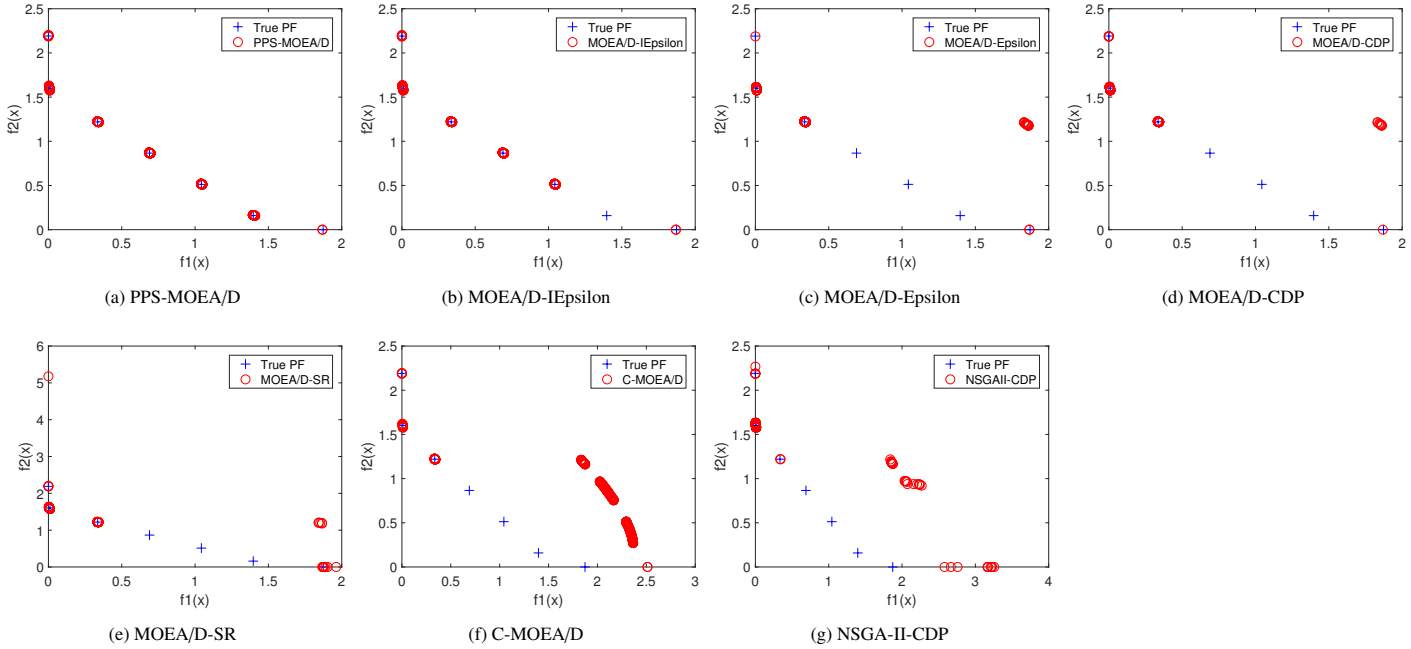
Figure 5: The non-dominated solutions achieved by each algorithm on LIR-CMOP6 with the median HV values.



(a) PPS-MOEA/D (b) MOEA/D-IEpsilon (c) MOEA/D-Epsilon (d) MOEA/D-CDP

(e) MOEA/D-SR (f) C-MOEA/D (g) NSGA-II-CDP

Figure 6: The non-dominated solutions achieved by each algorithm on LIR-CMOP7 with the median HV values.



(a) PPS-MOEA/D (b) MOEA/D-IEpsilon (c) MOEA/D-Epsilon (d) MOEA/D-CDP

(e) MOEA/D-SR (f) C-MOEA/D (g) NSGA-II-CDP

Figure 7: The non-dominated solutions achieved by each algorithm on LIR-CMOP11 with the median HV values.

the population in PPS-MOEA/D is lost, and it is difficult to pull the population to the whole true PF. From the experimental results, we can find that PPS-MOEA/D is not suitable for solving CMOPs which have unconstrained PFs with only one point.

5. Robot Gripper Optimization

In this section, a real-world optimization problem—the robot gripper optimization problem is formulated. Then, the proposed PPS-MOEA/D and the other six CMOEAs are tested on this optimization problem.

5.1. The formulation of the robot gripper optimization

The robot gripper optimization problem has two objectives and eight constraints, which is taken from [34]. The second and the fourth objectives of the original problem are used to formulate the robot gripper optimization problem in this work, while the constraints and the ranges of decision variables are kept the same to those in [34]. In this paper, the first objective $f_1(x)$ represents a force transmission ratio between the actuating force and the minimum gripping force. We prefer to transform more actuating force into the gripper force. Thus, this objective should be minimized. The second objective $f_2(x)$ is the sum of all elements of the robot gripper. It is relevant to the weight of the robot gripper, and minimizing $f_2(x)$ can lead to a lightweight design.

To study the characteristics of the robot gripper optimization problem, 3,000,000 solutions are generated as shown in Fig. 9, where 1,500,000 solutions are generated randomly, and the other 1,500,000 solutions are generated by MOEA/D-IEpsilon. We can observe that the two objectives are in conflict with each other.

5.2. Experimental study

5.2.1. Experimental settings

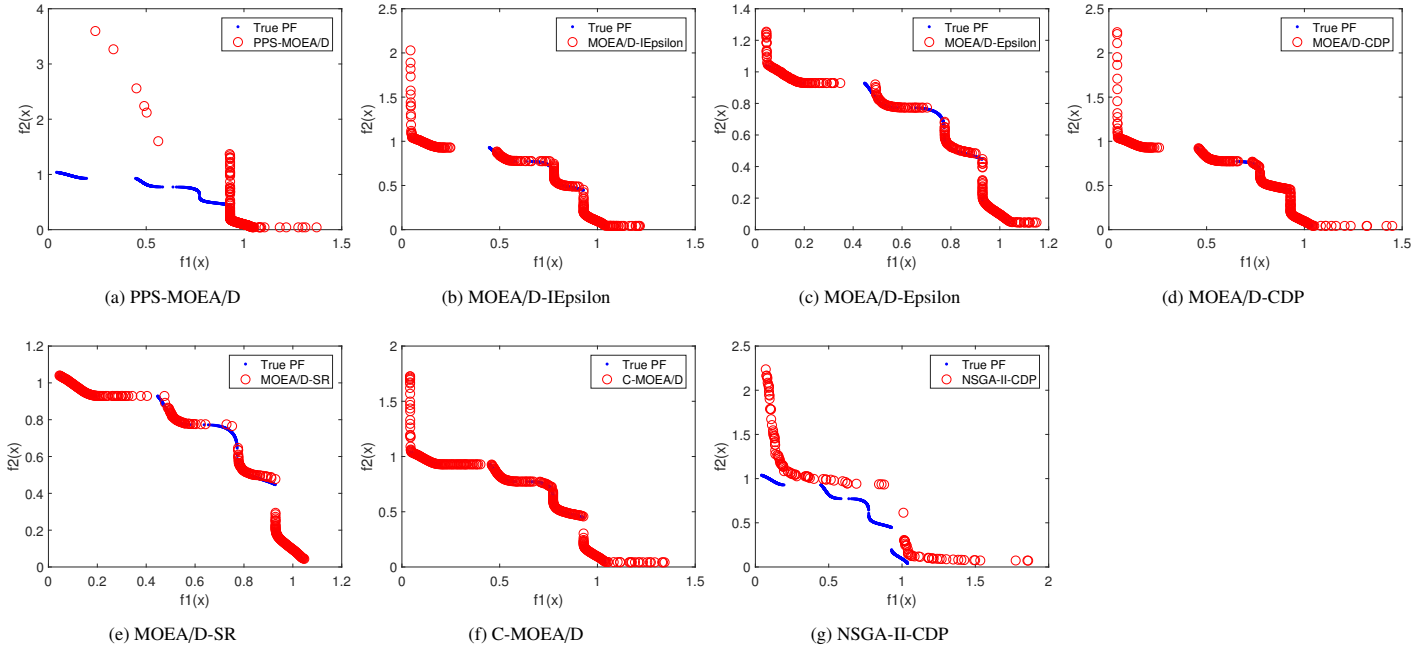
Five CMOEAs, including PPS-MOEA/D, MOEA/D-IEpsilon [22], MOEA/D-Epsilon [23], MOEA/D-SR [24], MOEA/D-CDP [24], C-MOEA/D [25] and NSGA-II-CDP [10], are tested on the robot gripper optimization problem. The parameters of these five CMOEAs are the same as listed in Section 4.1 except for the termination conditions. In the robot gripper optimization problem, each CMOEA stops when 600,000 function evaluations are reached. Since the true PF of the robot gripper optimization problem is unknown, We use the hypervolume metric [32] to measure the performance of these five CMOEAs, and the reference point is set to $[5, 800]^T$.

5.2.2. Analysis of experiments

The statistical results of *HV* values achieved by PPS-MOEA/D and the other six CMOEAs are shown in Table 6. We can observe that PPS-MOEA/D is better or significantly better than the other six CMOEAs. The non-dominated solutions achieved by each CMOEA with the median HV values during the 30 independent runs are plotted in Fig. 10(a)-(g). It is clear that PPS-MOEA/D has better or significantly better performance than the other six CMOEAs.

Three representative individuals (A, B and C) are selected from the non-dominated solutions achieved by PPS-MOEA/D with the best *HV* value in the 30 independent runs. The configurations of the robot gripper mechanism at each point are plotted in Fig. 11. We can observe that $f_1(\mathbf{x})$ is increasing with the decreasing of $f_2(\mathbf{x})$ from A to C, and each individual has a different geometrical structure.

The non-dominated solutions achieved by each algorithm on TNK-v1 with the median HV values.



(a) PPS-MOEA/D (b) MOEA/D-IEpsilon (c) MOEA/D-Epsilon (d) MOEA/D-CDP

(e) MOEA/D-SR (f) C-MOEA/D (g) NSGA-II-CDP

Figure 8: The non-dominated solutions achieved by each algorithm on TNK-v1 with the median HV values.

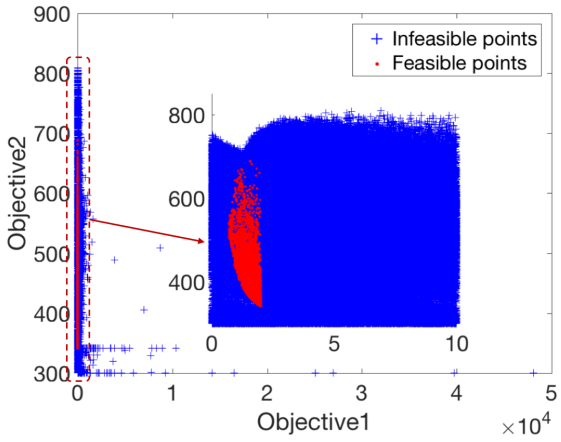
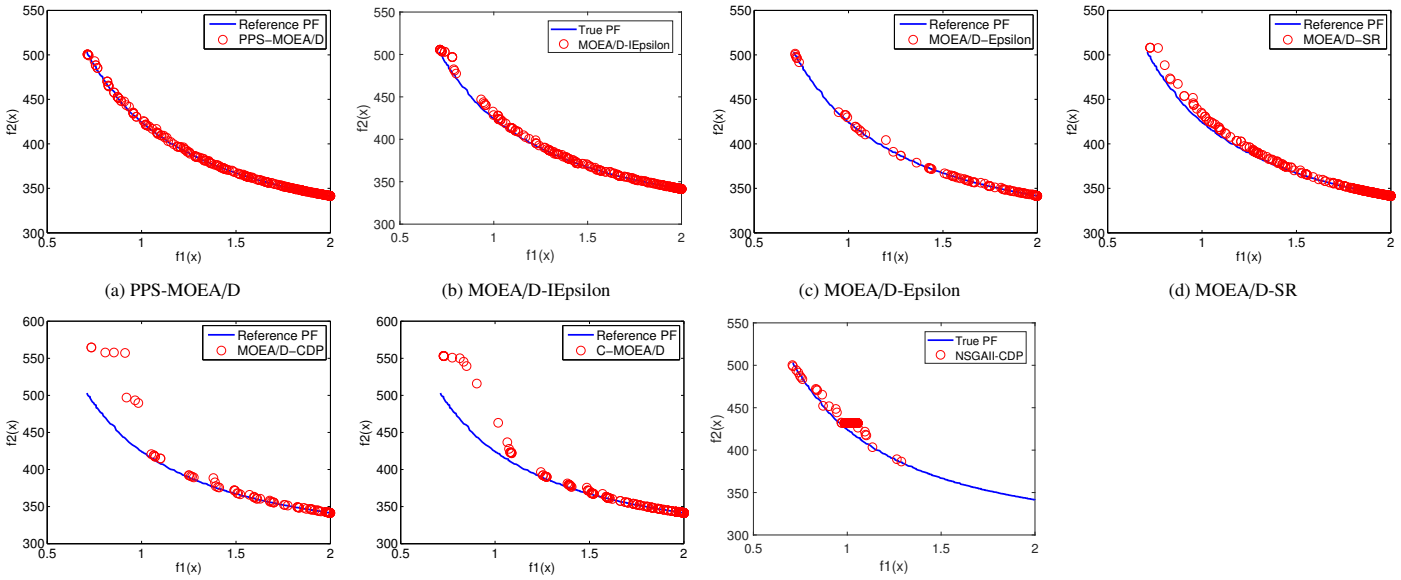


Figure 9: The distribution of solutions of the robot gripper optimization problem in the objective space.

6. Conclusion

This paper proposes a general PPS framework to deal with CMOPs. More specifically, the search process of PPS is divided into two stages—namely, push and pull search processes. At the push stage, constraints are ignored, which can help PPS to cross infeasible regions in front of the unconstrained PF. Moreover, the landscape affected by constraints can be estimated during the push stage, and this information, such as the ratio of feasible to infeasible solutions and the maximum overall constraint violation, can be applied to conduct the settings of parameters coming from the constraint-handling mechanisms in the pull stage. When the max rate of change between ideal and nadir points is less or equal than a predefined threshold, PPS is switched to the pull search process. The infeasible solutions achieved in the push stage are pulled to the feasible and non-dominated area by adopting an improved epsilon constraint-handling technique. The value of epsilon level can be set properly according to the maximum overall constraint violation obtained at the end of the push search stage. The comprehensive experiments indicate that the proposed PPS-MOEA/D achieves significantly better results than the other six CMOEAs on most of the benchmark problems and the robot gripper optimization problem.

It is also worthwhile to point out that there has been very little work regarding using information of landscape affected by constraints to solve CMOPs. In this context, the proposed PPS provides a viable framework. Obviously, a lot of work need to be done to improve the performance of PPS, such as, the strategy of searching around the borders between infeasible and feasible regions, the augmented constraint-handling mechanisms in the pull stage, the enhanced strategies to switch the search behavior, and the data mining methods and machine learning



(a) PPS-MOEA/D (b) MOEA/D-IEpsilon (c) MOEA/D-Epsilon (d) MOEA/D-SR

(e) MOEA/D-CDP (f) C-MOEA/D (g) NSGA-II-CDP

Figure 10: The non-dominated solutions achieved by each algorithm on the robot gripper optimization problem with the median HV values are plotted in (a)-(g). The reference PF consists of the non-dominated solutions achieved by each algorithm.

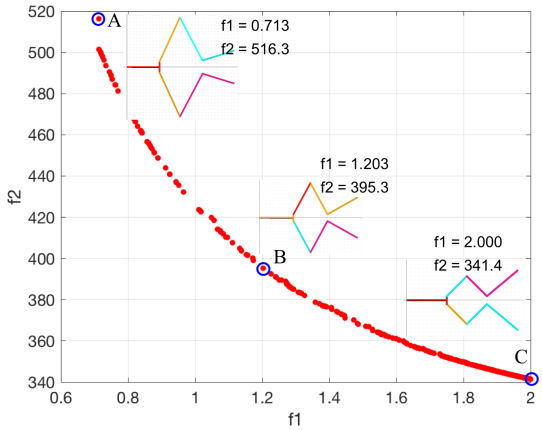


Figure 11: The non-dominated solutions achieved by PPS-MOEA/D with the best HV value in the 30 independent runs.

Table 3: IGD results of PPS-MOEA/D and PPS-MOEA/D1 on LIR-CMOP1-14. Wilcoxon's rank sum test at a 0.05 significance level is performed between PPS-MOEA/D and PPS-MOEA/D1. † and ‡ denote that the performance of MOEA/D1 is significantly worse than or better than that of PPS-MOEA/D, respectively. 'S-D-I' indicates PPS-MOEA/D is superior to, not significantly different from or inferior to PPS-MOEA/D1.

Test Instance		PPS-MOEA/D	PPS-MOEA/D1
LIRCMOP1	mean	**6.4134E-03**	1.8636E-02†
	std	1.9376E-03	8.1958E-03
LIRCMOP2	mean	**4.6730E-03**	1.2806E-02†
	std	7.8443E-04	5.2752E-03
LIRCMOP3	mean	**8.5450E-03**	4.8558E-02†
	std	5.1844E-03	1.5944E-02
LIRCMOP4	mean	**4.6773E-03**	2.9016E-02†
	std	1.1162E-03	9.3604E-03
LIRCMOP5	mean	1.8366E-03	**1.4714E-03**‡
	std	9.2633E-05	4.7991E-05
LIRCMOP6	mean	2.4895E-03	**1.4055E-03**‡
	std	3.3988E-04	3.0117E-05
LIRCMOP7	mean	**2.7972E-03**	2.9488E-03†
	std	9.8535E-05	1.1190E-04
LIRCMOP8	mean	**2.7778E-03**	2.8915E-03†
	std	7.5578E-05	5.3768E-05
LIRCMOP9	mean	9.9401E-02	**1.7489E-03**‡
	std	1.5187E-01	6.3562E-05
LIRCMOP10	mean	2.1081E-03	**1.8916E-03**‡
	std	7.7537E-05	6.8443E-05
LIRCMOP11	mean	**2.8318E-03**	4.3374E-03
	std	1.3587E-03	4.5296E-03
LIRCMOP12	mean	2.7035E-02	**2.4070E-03**
	std	5.0015E-02	4.5167E-04
LIRCMOP13	mean	6.4552E-02	**6.2769E-02**‡
	std	2.1770E-03	1.6426E-03
LIRCMOP14	mean	6.4186E-02	**6.3256E-02**
	std	1.6896E-03	1.2518E-03
Wilcoxon-Test (S-D-I)		–	6-3-5

Table 4: HV results of PPS-MOEA/D and PPS-MOEA/D1 on LIR-CMOP1-14. Wilcoxon's rank sum test at a 0.05 significance level is performed between PPS-MOEA/D and PPS-MOEA/D1. † and ‡ denote that the performance of MOEA/D1 is significantly worse than or better than that of PPS-MOEA/D, respectively. 'S-D-I' indicates PPS-MOEA/D is superior to, not significantly different from or inferior to PPS-MOEA/D1.

Test Instance		PPS-MOEA/D	PPS-MOEA/D1
LIRCMOP1	mean	**1.0157E+00**	1.0035E+00†
	std	1.5800E-03	7.1734E-03
LIRCMOP2	mean	**1.3492E+00**	1.3383E+00†
	std	1.0095E-03	6.0801E-03
LIRCMOP3	mean	**8.7030E-01**	8.3638E-01†
	std	2.6504E-03	1.5086E-02
LIRCMOP4	mean	**1.0927E+00**	1.0653E+00†
	std	2.4669E-03	1.1221E-02
LIRCMOP5	mean	1.4616E+00	**1.4624E+00**‡
	std	2.9194E-04	3.0454E-04
LIRCMOP6	mean	1.1286E+00	**1.1295E+00**‡
	std	1.7710E-04	1.1887E-04
LIRCMOP7	mean	**3.0151E+00**	3.0148E+00
	std	2.6625E-03	3.6326E-03
LIRCMOP8	mean	**3.0166E+00**	3.0160E+00†
	std	1.1394E-03	8.2996E-04
LIRCMOP9	mean	3.5696E+00	**3.7140E+00** ‡
	std	2.2415E-01	2.3984E-04
LIRCMOP10	mean	3.2410E+00	**3.2417E+00** ‡
	std	3.0767E-04	2.6240E-04
LIRCMOP11	mean	**4.3897E+00**	4.3896E+00
	std	2.2165E-04	2.3477E-04
LIRCMOP12	mean	5.6135E+00	**5.6884E+00** ‡
	std	1.5251E-01	8.4283E-05
LIRCMOP13	mean	**5.7100E+00**	5.6993E+00†
	std	1.2748E-02	1.0445E-02
LIRCMOP14	mean	**6.1930E+00**	6.1778E+00†
	std	1.3097E-02	1.1285E-02
Wilcoxon-Test (S-D-I)		–	7-2-5

Table 5: IGD and HV results of PPS and the other six CMOEAs on TNK-v1. To facilitate the display of this table, PPS, IEpsilon, Epsilon, CDP, and SR are short for MOEA/D-PPS, MOEA/D-IEpsilon, MOEA/D-Epsilon, MOEA/D-CDP, and MOEA/D-SR respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between PPS-MOEA/D and each of the other six CMOEAs. † and ‡ denote that the performance of the corresponding algorithm is significantly worse than or better than that of PPS-MOEA/D, respectively.

Test problem: TNK-v1		PPS	IEpsilon	Epsilon	SR	CDP	C-MOEA/D	NSGA-II-CDP
IGD	mean	2.69E-01	5.20E-02‡	1.33E-01‡	4.13E-03‡	1.28E-02‡	**4.03E-03**‡	1.89E-01‡
	std	8.28E-02	1.02E-01	1.44E-01	4.53E-04	2.98E-03	5.30E-04	8.11E-02
HV	mean	4.03E-01	7.00E-01‡	5.88E-01‡	7.57E-01‡	7.52E-01‡	**7.57E-01**‡	4.52E-01
	std	9.69E-02	1.31E-01	1.91E-01	5.49E-04	5.60E-04	5.49E-04	9.86E-02

Table 6: HV results of PPS-MOEA/D and the other six CMOEAs on the gripper optimization problem. To facilitate the display of this table, PPS, IEpsilon, Epsilon, CDP, and SR in this table are short for MOEA/D-PPS, MOEA/D-IEpsilon, MOEA/D-Epsilon, MOEA/D-CDP, and MOEA/D-SR respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between PPS-MOEA/D and each of the other six CMOEAs. † and ‡ denote that the performance of the corresponding algorithm is significantly worse than or better than that of PPS-MOEA/D, respectively. The best mean is highlighted in boldface.

Gripper optimization	PPS	IEpsilon	Epsilon	SR	CDP	C-MOEA/D	NSGA-II-CDP
mean	**1.895E+03**	1.893E+03	1.885E+03†	1.864E+03†	1.888E+03†	1.865E+03†	1.742E+03†
std	1.046E+01	4.845E+00	1.342E+01	1.134E+01	8.070E+00	9.435E+00	6.055E+01

approaches integrated in the PPS framework. For another future work, the proposed PPS will be implemented in the non-dominated framework, such as NSGA-II, to further verify the effect of PPS. More other CMOPs and real-world optimization problems will also be used to test the performance of the PPS embedded in different MOEA frameworks.

Acknowledgement

This research work was supported by the Key Lab of Digital Signal and Image Processing of Guangdong Province, the National Natural Science Foundation of China under Grant (61175073, 61300159, 61332002, 51375287), the Natural Science Foundation of Jiangsu Province of China under grant SBK2018022017, China Postdoctoral Science Foundation under grant 2015M571751, and Project of International, as well as Hongkong, Macao&Taiwan Science and Technology Cooperation Innovation Platform in Universities in Guangdong Province (2015KGJH2014).

Reference

[1] K. Deb, Multi-objective optimization using evolutionary algorithms, Vol. 16, John Wiley & Sons, 2001.

[2] C. A. C. 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 191 (11–12) (2002) 1245–1287.

[3] T. BDack, F. Hoffmeister, H. Schwefel, A survey of evolution strategies, in: Proceedings of the 4th international conference on genetic algorithms, 1991, pp. 2–9.

[4] A. Homaifar, C. X. Qi, S. H. Lai, Constrained optimization via genetic algorithms, Simulation 62 (4) (1994) 242–253.

[5] J. A. Joines, C. R. Houck, On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's, in: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, IEEE, 1994, pp. 579–584.

[6] J. C. Bean, A. ben Hadj-Alouane, A dual genetic algorithm for bounded integer programs, 1993.

[7] D. W. Coit, A. E. Smith, D. M. Tate, Adaptive penalty methods for genetic optimization of constrained combinatorial problems, INFORMS Journal on Computing 8 (2) (1996) 173–182.

[8] A. Ben Hadj-Alouane, J. C. Bean, A genetic algorithm for the multiple-choice integer program, Operations research 45 (1) (1997) 92–101.

[9] Y. G. Woldesenbet, G. G. Yen, B. G. Tessema, Constraint handling in multiobjective evolutionary optimization, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 514–525. doi:10.1109/TEVC.2008.2009032.

[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197. doi:10.1109/4235.996017.

[11] T. Takahama, S. Sakai, Constrained optimization by ε constrained particle swarm optimizer with ε-level control, in: Soft Computing as Transdisciplinary Science and Technology, Springer, 2005, pp. 1019–1029.

[12] T. P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Transactions on evolutionary computation 4 (3) (2000) 284–294.

[13] Z. Fan, W. Li, X. Cai, K. Hu, H. Lin, H. Li, Angle-based constrained dominance principle in moea/d for constrained multi-objective optimization problems, in: IEEE Congress on Evolutionary Computation, 2016, pp. 460–467.

[14] B. Li, J. Li, K. Tang, X. Yao, Many-objective evolutionary algorithms: A survey, ACM Computing Surveys (CSUR) 48 (1) (2015) 13.

[15] E. Mezura-Montes, C. A. Coello Coello, Constraint-handling in nature-inspired numerical optimization: Past, present and future, Swarm and Evolutionary Computation 1 (4) (2011) 173–194.

[16] Z. Cai, Y. Wang, A multiobjective optimization-based evolutionary algorithm for constrained optimization, Evolutionary Computation, IEEE Transactions on 10 (6) (2006) 658–675.

[17] T. Ray, H. K. Singh, A. Isaacs, W. Smith, Infeasibility driven evolutionary algorithm for constrained optimization, in: Constraint-handling in evolutionary optimization, Springer, 2009, pp. 145–165.

[18] S. Zeng, R. Jiao, C. Li, X. Li, J. S. Alkasassbeh, A general framework of dynamic constrained multiobjective evolutionary algorithms for constrained optimization, IEEE Transactions on Cybernetics 47 (9) (2017) 2678–2688. doi:10.1109/TCYB.2017.2647742.

[19] Y. Wang, Z. Cai, Y. Zhou, W. Zeng, An adaptive tradeoff model for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 80–92.

[20] B. Y. Qu, P. N. Suganthan, Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods, Engineering Optimization 43 (4) (2011) 403–416.

[21] K. Deb, An efficient constraint handling method for genetic algorithms, Computer methods in applied mechanics and engineering 186 (2) (2000) 311–338.

[22] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. You, J. Mo, C. Wei, E. D.

Goodman, An improved epsilon constraint-handling method in MOEA/D for cmops with large infeasible regions, arXiv preprint arXiv:1707.08767.
[23] Z. Yang, X. Cai, Z. Fan, Epsilon constrained method for constrained multiobjective optimization problems - some preliminary results., GECCO.
[24] M. A. Jan, R. A. Khanum, A study of two penalty-parameterless constraint handling techniques in the framework of MOEA/D, Applied Soft Computing 13 (1) (2013) 128–148.
[25] M. Asafuddoula, T. Ray, R. Sarker, K. Alam, An adaptive constraint handling approach embedded MOEA/D, in: 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–8.
[26] K. Miettinen, Nonlinear Multiobjective Optimization, Vol. 12, Springer Science & Business Media, 1999.
[27] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on evolutionary computation.
[28] T. Takahama, S. Sakai, Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites, in: 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 1–8. doi:10.1109/CEC.2006.1688283.
[29] H. Jain, K. Deb, An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II - Handling Constraints and Extending to an Adaptive Approach., IEEE Trans. Evolutionary Computation 18 (4) (2014) 602–622.
[30] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2) (2000) 311 – 338. doi:10.1016/S0045-7825(99)00389-8.
[31] P. A. Bosman, D. Thierens, The balance between proximity and diversity in multiobjective evolutionary algorithms, Evolutionary Computation, IEEE Transactions on 7 (2) (2003) 174–188.
[32] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.
[33] M. Tanaka, H. Watanabe, Y. Furukawa, T. Tanino, Ga-based decision support system for multicriteria optimization, in: 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, Vol. 2, 1995, pp. 1556–1561 vol.2. doi:10.1109/ICSMC.1995.537993.
[34] R. Saravanan, S. Ramabalan, N. G. R. Ebenezer, C. Dharmaraja, Evolutionary multi criteria design optimization of robot grippers, Applied Soft Computing 9 (1) (2009) 159–172.