An Evolution Path-Based Reproduction Operator for Many-Objective Optimization

Xiaoyu He, Yuren Zhou, and Zefeng Chen

***Abstract*—The many-objective evolutionary algorithms generally make use of a set of well-spread reference vectors to increase the selection pressure toward the Pareto front in high-dimensional objective space. However, few studies have been reported on how to generate new solutions toward the Pareto set (PS) in the decision space with the help of these reference vectors. To fill this gap, we develop a novel reproduction operator based on the differential evolution. The main idea is using the evolution paths to depict the population movement and predict its tendency. These evolution paths are used to create potential solutions, and thus, accelerate the convergence toward the PS. Furthermore, a self-adaptive mechanism is introduced to adapt related parameters automatically. This operator is implemented in two well-known many-objective evolutionary algorithm frameworks. The experimental results on 20 widely used benchmark problems show that the proposed operator is able to strengthen the performance of the original algorithms in handling many-objective optimization problems.**

***Index Terms*—Evolution path, many-objective optimization, parameter self-adaptation, reproduction operator.**

I. Introduction

MULTIOBJECTIVE optimization problems (MOPs) are problems that involve more than one conflicting objective functions. In this paper, we consider the following continuous MOP with box constraints:

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x})), \ldots, f_M(\boldsymbol{x}))^T \\ &\text{subject to} \quad \boldsymbol{x} \in \Omega \end{aligned}$$

where $\Omega = \prod_{i=1}^{D} [lb_i, ub_i] \subset R^D$ is the decision space, $\boldsymbol{F} : \Omega \rightarrow R^M$ consists of M real-value objective functions, and R^M is called the objective space. Unlike single objective optimization, the target of solving an MOP is to find a set of tradeoff solutions known as Pareto set (PS) in the decision space and Pareto front (PF) in the objective space, respectively. In the past two decades, a lot of multiobjective evolutionary algorithms (MOEAs) have been proposed to solve MOPs. These MOEAs have inherent superiority over classical methods on solving MOPs due to the capacity of finding a set of Pareto optimal solutions in a single run. One of the most famous approaches is to rank the solutions using the dominance relationship. Also, an explicit diversity management mechanism is required to preserve the population diversity. Remarkable examples of these Pareto-based MOEAs include PESA-II [1], NSGA-II [2], and SPEA2 [3].

MOEA based on decomposition (MOEA/D) [4] provides an alternative method by decomposing the MOP into a number of single-objective optimization subproblems. Each subproblem is then optimized in a collaborative manner using the evolutionary algorithm. As demonstrated by massive studies, MOEA/D is very competitive with Pareto-based algorithms especially when dealing with problems having regular PFs [5].

Showing excellent performance in solving MOPs, the above MOEAs are faced with difficulties when tackling MOPs with more than three objectives [6], [7]. These MOPs are referred to as many-objective optimization problems (MaOPs). The main challenge brought by MaOPs is the deterioration of selection pressure due to the increasing number of objectives. Though originally designed for solving MOPs, the decomposition strategy in MOEA/D provides a promising framework to design new many-objective evolutionary algorithms (MaOEAs). One method is to divide the objective space into multiple subregions. Then, an aggregation function is adopted to select solutions in each subregion. MaOEAs using this method includes RVEA [8] and MOEA/D-DU [9]. Alternatively, since each subregion contains only a small portion of solutions, the selection pressure is increased, and thus, conventional Pareto-based methods such as nondominated sorting [10] can be employed to select solutions. For example, NSGA-III [11] first classifies the population into different nondominated levels, and then, selects solutions from the last acceptable level with a niche-preservation operator. It is worth noting that though NSGA-III does not explicitly decompose the MaOP like MOEA/D, it implicitly decomposes the objective space in such a way that diversity of the solutions close to every possible reference point is attempted to be maintained [12]. Therefore, NSGA-III belongs to the decomposition-based algorithms in essence. Other algorithms such as MOEA/D-M2M [13], MOEA/DD [14], and MOEA/PIE [15] also share

Manuscript received July 14, 2017; revised October 7, 2017 and December 13, 2017; accepted December 14, 2017. Date of publication December 19, 2017; date of current version January 28, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61472143, Grant 61773410, and Grant 61673403, and in part by the Scientific Research Special Plan of Guangzhou Science and Technology Programme under Grant 201607010045. *(Corresponding author: Yuren Zhou.)*

The authors are with the School of Data and Computer Science and the Collaborative Innovation Center of High Performance Computing, Sun Yat-sen University, Guangzhou 510006, China (e-mail: hxyokokok@foxmail.com; zhouyuren@mail.sysu.edu.cn; chzfeng@mail2.sysu.edu.cn).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2017.2785224

similar characteristics. As we know, all of the above algorithms share the following common features.

1) A set of well-spread reference vectors is employed in the selection operator to increase the selection pressure toward the PF.
2) New solutions are generated in a randomized way. In fact, nearly all the MaOEAs use simulated binary crossover (SBX) [16], [17] to generate new solutions. This is a randomized method without using search-direction information.

In other word, when generating a new solution in the decision space, the reference vectors are not utilized to guide the searching process toward the PS. Some recently proposed $R2$ indicator-based algorithms like MOMBI [18] and MOMBI-II [19] also have the above features. A question then arises naturally: is there a useful way to generate new solutions toward the PS with the help of a set of reference vectors? To answer this question, we are faced with the following issues.

1) The set of reference vectors is defined in the objective space and it is difficult to utilize the reference vectors in the decision space.
2) Most reproduction operators involve some parameters and their best settings are usually problem-dependent. According to the no free lunch theorem [20], setting them to fixed values is not suitable for solving all problems. Consequently, it is desirable to design a parameter self-adaptation mechanism to adapt these parameters automatically.

To address the above issues and to design a new reproduction operator, we first review some popular reproduction operators. In evolutionary algorithms, the methods of generating new candidate solutions can be divided into two categories: solution-based approach and model-based approach. In the solution-based approach, a solution is generated directly from existed solutions under specific rules. One of the early implementations of this approach is the blend crossover (BLX-α) [21]. In this operator, offspring solutions are uniformly sampled near the parents, and their extent is in proportion to the parent solutions. SBX further extends BLX-α by assigning more probability for creating solutions near their parents. This makes SBX especially useful in MaOEAs since the population distribution will not be changed severely after obtaining offspring solutions. Particle swarm optimization (PSO) [22] is one of the most famous bio-inspired operators in this category [23]. Unlike BLX-α and SBX, it utilizes the promising solutions to guide the search, thereby showing high convergence rate. Differential evolution (DE) [24] is another popular solution-based operator. It generates offspring solutions by perturbing the population solutions with the scaled differences of randomly selected parents. Thus, it does not require extra probability distribution to explore the function landscape as BLX-α and SBX do. Sindhya *et al.* [25] considered the offspring generated by DE as a linear combination of its parents. They further suggested to generate offspring through polynomial-based interpolation with the parent solutions. A hybrid operator (HOP) is then proposed to hybridize the polynomial-based interpolation with the original DE, and shows clear performance advantages. Note that, though DE and PSO have been widely used in MOEAs [26]–[30], the experiments in [11] and [31] have shown that they may not be suited for solving MaOPs. In summary, the above solution-based operators are simple and efficient, but it is difficult to steer their search when a reference vector is given in the objective space.

Whereas, in the model-based approach, a model is built first and new solutions are generated using this model. Estimation of distribution algorithms (EDAs) are in this category. In the past decade, a number of multiobjective EDAs have been proposed, and shown a promising prospect in handling MOPs and even MaOPs [32]–[35]. In these algorithms, the objective space is divided into small subregions using the reference vectors. Then, a probability model in the decision space is built for each subregion using the corresponding solutions. Since these models are updated iteratively, they are able to depict the population distribution and predict the movement tendency toward the PS. In this way, the reference vectors are utilized to guide the search in the decision space. However, the statistical learning techniques embedded in these algorithms are so complicated that building the probability model is time-consuming. Generally, these model-based reproduction operators indeed provide a way to utilize the reference vectors in the decision space, but it suffers from low efficiency.

Apart from the field of evolutionary algorithms, the topic of how to search with a given reference vector has been studied for many years in the mathematical programming community. The related methods are known as "objective steering methods." Normal-boundary intersection [36], which converts an MOP into a series of goal programming problems, is a classic method in early works. Other remarkable methods include combined-objectives repeated line search [37] and directed search [38]. These methods are helpful in increasing the convergence speed compared with traditional stochastic search methods. Their limitation is that the gradient information of the objective functions are usually required. When these information cannot be obtained, extra function evaluations or the solution neighborhood structure are required to estimate the gradient. Also, as demonstrated by the experiments in [37] and [39], using the above methods alone is not likely to provide satisfactory results. Thus, the hybridization with traditional evolutionary algorithms is essential. For example, adaptive resource allocation [40] can be employed to determine the probability of performing these methods. Another approach is to apply different search strategies in different regions in the objective space [41]. Nevertheless, how to design a proper hybridization strategy is still an open issue.

To take the advantages of all the above approaches, an evolution path-based DE operator is proposed in this paper. Evolution path is first introduced in covariance matrix adaptation evolution strategy (CMA-ES) [42], [43], and it is a model describing the path the population takes over a number of generations. Usually, it has some significant statistical characteristics, and thus can be utilized to depict the population distribution. For example, both CMA-ES and multiobjective CMA-ES [44] employ evolution paths to calculate the unbiased estimators of the population covariance matrix. But as mentioned before, calculating multiple covariance matrices

seems to be unpractical in MaOEAs due to its time complexity. The DE with an evolution path framework (DEEP) [45] provides a much simpler way of utilizing evolution path to improve convergence rate of the DE. However, this method is only designed for single-objective optimizations.

In this paper, we propose a novel way to utilize evolution paths without increasing time complexity. First, each reference vector is associated with a set of evolution paths. A potential solution is calculated using the set of evolution paths by means of extrapolation. This potential solution is then injected into a self-adaptive DE operator to produce an offspring solution. Finally, after obtaining the offspring and performing the environmental selection, the corresponding evolution paths are updated by the solutions in the new population. In this way, the evolution paths are able to trace the solution curves for each reference vector and can be utilized to accelerate the convergence. Moreover, if the reference vectors are well-spread, the proposed operator also facilitates maintaining the wideness and uniformity of the offspring solutions.

The contributions of this paper can be summarized as follows.

1) A novel method of constructing the evolution paths and utilizing them to generate potential solutions has been proposed. We also provide a delicate way to inject these potential solutions into DE operators in order to guide the search. To the best of our knowledge, no work has been reported on using evolution path in MaOEAs.
2) A self-adaptive mechanism has been introduced to adapt the parameters automatically. We implement a data structure to store successful parameters (i.e., parameters yielding promising offspring in recent generations). The new parameters are generated by means of sampling parameter space close to the stored ones.
3) The proposed operator has been implemented in two well-known decomposition-based MaOEAs. The experimental results demonstrate that this new operator improves the performance of MaOEAs in terms of two widely used performance indicators.

The remainder of this paper is organized as follows. Section II discusses the basic idea of the newly designed operator. Section III describes the detailed implementations in two state-of-the-art MaOEAs. The numerical experiments are presented in Section IV. Section V concludes this paper.

II. Proposed Reproduction Operator

In this section, we first provide the basic ideas of the proposed operator. Then, the main components of the proposed operator are described in detail.

A. Basic Ideas

To facilitate our explanation in this section, MOEA/D is used as the algorithm framework. It decomposes an MOP into multiple subproblems with the help of a set of reference vectors. Each subproblem maintains a certain solution which is usually the best solution found so far for this subproblem. In the typical implementation of MOEA/D, once a better solution is found, the former solution is discarded and replaced by the new one. However, it is very likely that this discarded solution may also provide useful information to guide the search. Therefore, our idea is to collect the historical information when the above replacement happens. Then, we can predict potential solutions using these information to increase the convergence rate.

One essential step in the above idea is to collect the historical information for a given subproblem. In this paper, this is done by using the evolution paths. An evolution path is a vector defined in the decision space Ω. We calculate it within an iterative process by weighted summation of newly obtained solutions. Formally, we denote $\boldsymbol{ep}_i^{(g)}$ and $\boldsymbol{u}^{(g)}$ as the evolution path and the newly obtained solution, respectively, for the ith subproblem after the gth replacement happens. Then, the evolution path is recursively defined as follows:

$$\begin{aligned} \boldsymbol{ep}_i^{(g)} &= (1-c)\cdot \boldsymbol{ep}_i^{(g-1)} + c\cdot \boldsymbol{u}^{(g)} \\ \boldsymbol{ep}_i^{(0)} &= \boldsymbol{u}^{(0)} \end{aligned} \tag{1}$$

where $c \in [0, 1]$ is a controlling factor and $\boldsymbol{u}^{(0)}$ denotes the initial solution for the ith subproblem.

According to (1), an evolution path is the weighted summation of all the historical solutions for a given subproblem, which makes it a reasonable method to describe the historical information. However, it is clear that the evolution path is determined by the controlling factor c. When c is fixed, one single evolution path can only describe a static state of the historical solutions. Remember that, our goal is to predict the potential solutions using the historical solutions, which means we have to extract some "dynamic" information. One straightforward approach is to construct multiple evolution paths with c being different values. For example, for the ith subproblem, $\boldsymbol{ep}_i^{(g)}$ describes its initial solution with $c = 0$ and the current solution with $c = 1$. Then, with some mathematical tools, we can build a relationship between $\boldsymbol{ep}_i^{(g)}$ and c. This relationship, considered as a kind of historical information, is finally utilized to generate potential solutions.

B. Constructing the Evolution Paths

This section shows how to construct the evolution paths in MOEA/D. Let $\{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_H\}$ be a set of H reference vectors. For each reference vector $\boldsymbol{\lambda_i}$, let $\{\boldsymbol{ep}_{i,1}, \boldsymbol{ep}_{i,2}, \ldots, \boldsymbol{ep}_{i,S}\}$ be a set of D-dimensional vectors, where S is a positive integer, and $\boldsymbol{ep}_{i,t}$ is called the evolution path with the updating rate t.[1] These totally $S \cdot H$ evolution paths are stored in a table ***EPTable*** (shown in Fig. 1), where each entry contains an evolution path.

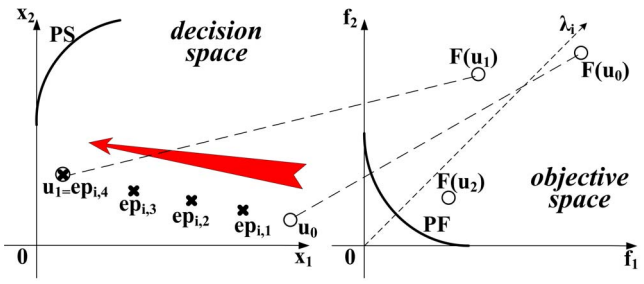
1) Initialization: In the beginning, ***EPTable*** are initialized using the initial population. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ be the initial population, where $\boldsymbol{x}_i \in \Omega$ is the ith solution and N is the population size. Then each evolution path in ***EPTable*** is set to a random solution

$$\boldsymbol{ep}_{i,t} = \boldsymbol{x}_{r_i},\ i = 1, 2, \ldots, H,\ t = 1, 2, \ldots, S \tag{2}$$

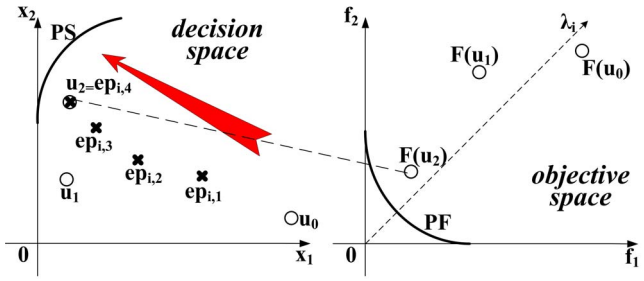
[1] In MOEA/D, the best solution corresponding to a given reference vector may be replaced multiple times in one generation. Therefore, the superscript g in (1) can be different values for different reference vectors. To simplify the following descriptions, the superscript g is omitted.

index	1	2	⋯	S
1	$ep_{1,1}$	$ep_{1,2}$	⋯	$ep_{1,S}$
2	$ep_{2,1}$	$ep_{2,2}$	⋯	$ep_{2,S}$
⋮	⋮	⋮	⋯	⋮
H	$ep_{H,1}$	$ep_{H,2}$	⋯	$ep_{H,S}$

Fig. 1. Illustration of the table ***EPTable***. Entries in the ith row contain the evolution paths associated with the reference vector $\boldsymbol{\lambda}_i$. Evolution paths stored in the tth column have the updating rate t.



(a) updating after $\boldsymbol{u}_0$ is replaced by $\boldsymbol{u}_1$



(b) updating after $\boldsymbol{u}_1$ is replaced by $\boldsymbol{u}_2$

Fig. 2. Updating the evolution paths. The thick curves describe the PF in the objective space and the PS in the decision space, respectively. The circles depict the solutions and the crosses depict the evolution paths. The red arrow describes the direction pointed from evolution path with a small updating rate to that with a large updating rate. Updating after (a) $\boldsymbol{u}_0$ is replaced by $\boldsymbol{u}_1$ and (b) $\boldsymbol{u}_1$ is replaced by $\boldsymbol{u}_2$.

where r_i in an integer randomly selected from $\{1, 2, \ldots, N\}$ for each i.

2) Updating: For each reference vector $\boldsymbol{\lambda}_i$, when its solution is replaced by a new solution $\boldsymbol{u}$, its corresponding evolution paths are updated as follows:

$$\boldsymbol{ep}_{i,t} = \left(1 - \frac{t}{S}\right) \cdot \boldsymbol{ep}_{i,t} + \frac{t}{S} \cdot \boldsymbol{u}, t = 1, 2, \ldots, S. \tag{3}$$

Fig. 2 illustrates how the evolution paths are changed in the first two replacements corresponding to the ith subproblem. Here, S is set to 4 and $\boldsymbol{u}_0$ is the initial solution. For convenience, the initial values of the four evolution paths are set to $\boldsymbol{u}_0$, as well. Suppose $\boldsymbol{u}_0$ is first replaced by $\boldsymbol{u}_1$. In this case, the four evolution paths are located on the line between $\boldsymbol{u}_0$ and $\boldsymbol{u}_1$ [shown in Fig. 2(a)]. After obtaining a new solution $\boldsymbol{u}_2$ which is able to replace $\boldsymbol{u}_1$, all the evolution paths move toward $\boldsymbol{u}_2$ [shown in Fig. 2(b)]. It is observed that when the solutions move toward the PF in the objective space, the corresponding evolution paths move toward the PS in the decision space at the same time. Also, the larger the updating rate is, the closer an evolution path gets to the PS. Consider the direction

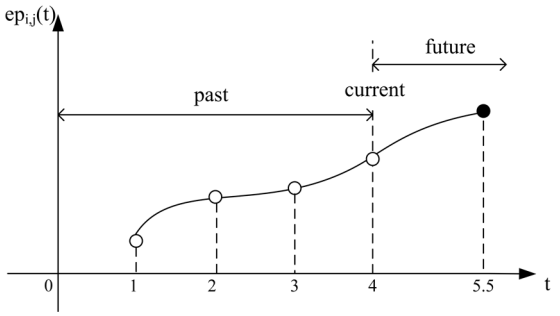


Fig. 3. Illustration of calculating the ith decision variable of a potential solution with $S = 4$. The empty circle represents the jth component of $\boldsymbol{ep}_{i,t}$ concerning a given t, and $t \in \{1, 2, 3, 4\}$ is a predefined value which means it is "in the past." The solid curve depicts the interpolation polynomial $\text{ep}_{i,j}(t)$. The solid circle shows a potential solution $\text{ep}_{i,j}(t^*)$, where $t^* = 5.5 > 4$ means it is "in the future."

pointing from the evolution path with a small updating rate to that with a large updating rate (i.e., from $\boldsymbol{ep}_{i,1}$ to $\boldsymbol{ep}_{i,4}$). As depicted by the red arrow, this direction also points to the PS. Moreover, this direction becomes more accurate after the replacement happens. In the next section, we will use this direction to accelerate the search process.

C. Generating Potential Solutions

Let $\boldsymbol{ep}_i(t) = (\text{ep}_{i,1}(t), \text{ep}_{i,2}(t), \ldots, \text{ep}_{i,D}(t))^T = \boldsymbol{ep}_{i,t}$. That is, we consider each component of $\boldsymbol{ep}_{i,t}$ as a function of t. Then, evolution path has some special properties.

Property 1: $\boldsymbol{ep}_i(0) = \boldsymbol{x}_{i,\text{init}}$, where $\boldsymbol{x}_{i,\text{init}}$ is the initial solution corresponding to $\boldsymbol{\lambda}_i$.

Property 2: $\boldsymbol{ep}_i(S) = \boldsymbol{x}_{i,\text{current}}$, where $\boldsymbol{x}_{i,\text{current}}$ is the current solution corresponding to $\boldsymbol{\lambda}_i$.

The above properties can be deduced directly from the initialization and updating rule of the evolution paths. Moreover, from Fig. 2, we have the following observations.

Observation 1: If $0 < t_0 < S$, $\boldsymbol{ep}_i(t_0)$ is somewhere between $\boldsymbol{x}_{i,\text{init}}$ and $\boldsymbol{x}_{i,\text{current}}$.

Observation 2: If $0 < t_1 < t_2 < S$, $\boldsymbol{ep}_i(t_2)$ gets closer to $\boldsymbol{x}_{i,\text{current}}$ than $\boldsymbol{ep}_i(t_1)$ does.

With the help of evolution path, the time range of the search process is normalized to $[0, S]$: $\boldsymbol{ep}_i(0)$ denotes the initial solution, $\boldsymbol{ep}_i(S)$ denotes the current solution, and $\boldsymbol{ep}_i(t)$ with $t \in (0, S)$ indicates a position in the past history. The basic idea of using the evolution paths to predict a promising offspring is: If we are able to generate an $\boldsymbol{ep}_i(t^*)$ with $t^* > S$, it can be considered as a potential solution *in the future*. Details of generating this potential solution are as follows.

For each $j \in \{1, 2, \ldots, D\}$, we have S pairs of points: $(1, \text{ep}_{i,j}(1)), (2, \text{ep}_{i,j}(2)), \ldots, (S, \text{ep}_{i,j}(S))$. Based on these points, a Lagrange polynomial is constructed to approximate $\text{ep}_{i,j}(t)$ by means of equidistant interpolation

$$\begin{aligned} \text{ep}_{i,j}(t) &= \Sigma_{k=1}^{S} \beta_k(t) \cdot \text{ep}_{i,j}(k) \\ \beta_k(t) &= \prod_{1 \le r \le S, r \ne k} \frac{t - r}{k - r}. \end{aligned} \tag{4}$$

Now, with the help of the above polynomial, we are able to generate a potential solution $\boldsymbol{ep}_i(t^*)$ with $t^* > S$. Fig. 3

provides an example with $S = 4$ and $t^* = 5.5$. Note that, since t^* is beyond the range $[0, S]$, the extrapolation results may be inaccurate if the value of t^* or S is too large. Consequently, we limit $t^* < 2S$ and $S < 5$.

D. Combining Evolution Path With Differential Evolution

The potential solution $\boldsymbol{ep}_i(t^*)$ cannot be used as an offspring solution immediately since the parameter t^* is difficult to determined. Now we propose a method to combine it with a DE operator, which avoids choosing a proper value for t^*. Specifically, we can generate an offspring $\boldsymbol{u} = (u_1, u_2, \ldots, u_D)^T$ for each solution $\boldsymbol{x}_i$ as follows:

$$\boldsymbol{v} = \boldsymbol{ep}_i\left(S \cdot \left(1 + \frac{FEs}{\max FEs} \cdot (1 - F)\right)\right) + F \cdot \left(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_1}\right) \quad (5)$$

$$u_k = \begin{cases} v_k & \text{with probability } Cr \\ x_{i,k} & \text{with probability} 1 - Cr \end{cases}, \quad k = 1, 2, \ldots, D \quad (6)$$

where $\boldsymbol{x}_{r_1}$ and $\boldsymbol{x}_{r_2}$ are two neighbors randomly chosen from the mating pool, FEs is the number of consumed fitness evaluations, maxFEs is the maximum number of fitness evaluations allowed, and F and Cr are two parameters of DE operator.

Clearly, compared with the canonical DE operator, this operator searches the nearby region of $\boldsymbol{ep}_i(t^*)$ rather than that of $\boldsymbol{x}_i$. The reason is: the population is getting closer and closer to the true PF because of the environmental selection. According to the updating rule of evolution path, it is very likely that $\boldsymbol{ep}_i(t^*)$ is closer to the PS than $\boldsymbol{x}_i$ [due to the fact that $\boldsymbol{x}_i = \boldsymbol{ep}_i(S)$ and $S < t^*$]. Therefore, this scheme can accelerate the convergence along the direction of $\boldsymbol{\lambda}_i$.

It can also be observed from (5) that, we have set t^* to $S \cdot (1 + (FEs/\max FEs) \cdot (1 - F))$. The reasons are as follows.

1) F is the scaling parameter in DE operator which is usually inside the range $[0, 1]$. Consequently, the value of t^* is between S and $2S$. The lower boundary makes $\boldsymbol{ep}_i(t^*)$ a potential solution in the future, while the upper boundary prevents it from being far away from current solution.
2) In DE, F maintains the power of both exploitation (with a small F value) and exploration (with a large F value) [46]. The larger the value F is, the more randomness will be brought by the difference vector $(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_1})$. On the other hand, because $\boldsymbol{ep}_i(t^*)$ is a high-order polynomial, the extrapolation accuracy decreases with the increasing of t^*. In order to balance the extrapolation accuracy and the randomness, keeping t^* decreasing with the increasing F seems to be an ideal strategy.
3) t^* decreases as FEs increases. This design makes it possible that the convergence capability is emphasized in the early stage and decreased gradually throughout the search process. Besides, in the end of the search process, the proposed reproduction operator will be degenerated to the canonical DE operator described in [28].

E. Parameter Self-Adaptation

Due to the hybridization with DE, the proposed reproduction operator involves two parameters: 1) F and 2) Cr. As we have known, the performance of DE has a strong relationship with these parameters. Here, we propose a novel mechanism which is able to adapt these parameters automatically. This method is inspired by the success-history-based adaptive DE which has shown excellent performance in CEC competitions on single objective optimization [47]. In this part, we extend the existing work in handling MaOPs.

index	1	2	···	HL	pointer
hm_1	$\{F_{1,1}, Cr_{1,1}\}$	$\{F_{1,2}, Cr_{1,2}\}$	···	$\{F_{1,HL}, Cr_{1,HL}\}$	ptr_1
hm_2	$\{F_{2,1}, Cr_{2,1}\}$	$\{F_{2,2}, Cr_{2,2}\}$	···	$\{F_{2,HL}, Cr_{2,HL}\}$	ptr_2
⋮	⋮	⋮	···	⋮	⋮
hm_H	$\{F_{H,1}, Cr_{H,1}\}$	$\{F_{H,2}, Cr_{H,2}\}$	···	$\{F_{H,HL}, Cr_{H,HL}\}$	ptr_H

Fig. 4. Structure for storing parameters. The parameters are stored in a table where each row presents a history memory for a given reference vector. Newly generated parameters being successful in the evolution can be written to a certain position in the table pointed by the pointer.

As shown in Fig. 4, we maintain a historical memory $\boldsymbol{hm}_i$ with HL entries for each reference vector $\boldsymbol{\lambda}_i$, where $i = 1, 2, \ldots, H$. The kth of entry $\boldsymbol{hm}_i$ (denoted by $\boldsymbol{hm}_i[k]$) contains a pair of parameters $\{F_{i,k}, Cr_{i,k}\}$. We also maintain a pointer ptr_i (shown in the last column in Fig. 4) pointing to the writable position of $\boldsymbol{hm}_i$.

The parameter self-adaptive mechanism works as follows.

1) At the beginning, all parameters F and Cr are set to 0.5. All the pointers are set to 1.
2) In the reproduction stage, we use the following method to generate parameters for $\boldsymbol{\lambda}_j$:

$$F = \text{rand}n() \cdot 0.1 + F_{j,\text{jrand}}$$
$$Cr = \text{rand}n() \cdot 0.1 + Cr_{j,\text{jrand}} \quad (7)$$

   where $j$rand is an integer randomly chosen from $\{1, 2, \ldots, HL\}$ and rand$n()$ is a procedure which generates a random number sampling from a standard normal distribution. F and Cr are regenerated if their values are outside the range $[0, 1]$.
3) After the environmental selection, if an offspring solution survives from the selection operator and is associated with a reference vector $\boldsymbol{\lambda}_i$, its parameters are stored in the ith historical memory. More precisely, the parameters are written to the entry pointed by ptr_i (i.e., $\boldsymbol{hm}_i[\text{ptr}_i]$). Then, ptr_i points to the next entry. If the value of ptr_i is large than HL, ptr_i is set to 1.

Fig. 5 illustrates the above process. For a certain reference vector, the self-adaptation mechanism is able to generate new parameters close to the old parameters. If the new parameters are helpful in generating promising solutions, these solutions, in turn, help to deliver these parameters to nearby reference vectors. Moreover, when newly obtained parameters enter into a historical memory, the oldest ones are removed (i.e., the corresponding entry is overwritten) and will not influence the next steps. Generally, the above mechanism makes the parameters adapted gradually by learning from the previous experiences of generating promising offspring.

Now we have described all the components of the proposed reproduction operator. For convenience, we organize the above works as two procedures: 1) *reproduction* and 2) *adaptation*. The pseudo-codes are provided in Algorithms 1 and 2, respectively. Note that, the polynomial mutation is also performed

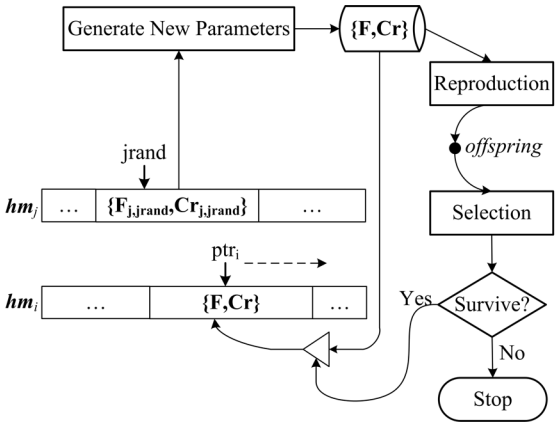


Fig. 5. Illustration of parameter self-adaptation. A pair of parameters are randomly chosen from $\boldsymbol{hm}_j$ to generate new parameters F and Cr using (7). F and Cr are employed in the reproduction operator to generate an offspring. If this offspring is able to survive from the selection and is associated with reference vector $\boldsymbol{\lambda}_i$, F, and Cr are written to the entry $\boldsymbol{hm}_i[\text{ptr}_i]$. Then, ptr_i points to the next entry.

Algorithm 1 Reproduction

Input: $\boldsymbol{x}$: Parent solution
P: Mating pool
$\boldsymbol{epSet} = \{\boldsymbol{ep}_{i,1}, \boldsymbol{ep}_{i,2}, \ldots, \boldsymbol{ep}_{i,S}\}$: A set of evolution paths
$\boldsymbol{hm}_i$: A historical memory of parameters
Output: An offspring solution and the parameters F and Cr
1: Generate F, Cr from $\boldsymbol{hm}_i$ according to Eq. (7)
2: Randomly select $\boldsymbol{x}_{r_1}, \boldsymbol{x}_{r_2}$ from P
3: Generate $\boldsymbol{u}$ according to Eq. (4) to (6)
4: $\boldsymbol{u} \leftarrow polynomialMutation(\boldsymbol{u})$
5: **return** $\{\boldsymbol{u}, F, Cr\}$

Algorithm 2 Adaptation

Input: $\boldsymbol{hm}_i$: A historical memory of parameters
ptr_i: A pointer
HL: Length of the historical memory
F, Cr: Successful parameters
1: $\boldsymbol{hm}_i[ptr_i] \leftarrow \{F, Cr\}$
2: $ptr_i \leftarrow ptr_i + 1$
3: **if** $ptr_i > HL$ **then**
4: $ptr_i \leftarrow 1$
5: **end if**

on the offspring as suggested in [28] since it slightly increases the algorithm performance.

F. Discussion

As mentioned before, the proposed reproduction operator is especially designed for handling MaOPs. This is mainly achieved by introducing the restricted recombination scheme. Thus, this section presents some discussions about why this scheme is helpful in high-dimensional objective space. Also, the proposed operator uses the extrapolation technique and parameter self-adaption, which shares some common features with objective steering methods mentioned in Section I. Their differences are also summarized in this section.

1) Restricted Recombination Scheme: It is known that in solving MaOPs, the proportion of the region that a solution dominates decreases exponentially with an increase of number of objectives. It causes a side effect that the nondominated solutions tend to cover a large portion of objective and decision space. If the recombination is performed between distant solutions, the results may be disruptive since the offspring is usually far away from its parents [11], [48].

The offspring obtained in the proposed operator can be considered as a recombination among four solutions: the parent solution, the potential solution produced using evolution paths, and two random solutions used in the DE operator. To alleviate the about issue, two restrictions have been put on this recombination. First, for each parent solution $\boldsymbol{x}_i$, only the evolution paths corresponding to the reference vector $\boldsymbol{\lambda}_i$ are used to generate the potential solution [i.e., $\boldsymbol{ep}_i(t^*)$]. Second, the two random solutions [i.e., $\boldsymbol{x}_{r_1}$ and $\boldsymbol{x}_{r_2}$ in (5)] are chosen from the restricted mating pool.[2] Thus, the offspring solution is likely to be close to the reference vector $\boldsymbol{\lambda}_i$.

In addition, the parameter self-adaption mechanism is also restricted. Take MOEA/D as an instance, an offspring solution can only replace those in its neighborhood. It means its parameters are only written to its neighbors' historical memories. In this way, solutions being close to each other usually have similar search steps, which helps to preserve the population diversity.

2) Comparisons With Objective Steering Methods: As described in Section I, the objective steering methods in the mathematical programming community also utilize the reference vectors to guide the search in the decision space. However, there are some essential differences between the proposed operator and the objective steering methods.

1) Objective steering methods generally construct a relationship between the objectives and the decision variables. Given a reference vector defined in the objective space, a corresponding "descent direction" in the decision space is then calculated to guide the search. On the contrary, the proposed operator tries to find out the relationship between the decision variables and the updating rates. With this relationship, a potential solution can be calculated directly through extrapolation. Therefore, there is no descent direction in the proposed operator.
2) Given a descent direction, an objective steering method requires to calculate a proper search step by using techniques such as line search [49]. In the proposed operator, the search step is controlled by the DE parameters which are automatically determined using the parameter self-adaption mechanism.
3) When calculating the descent direction in objective steering methods, the pair-wise dependences between decision variables are usually taken into consideration. It brings some computation-intensive tasks such as matrix decomposition and matrix inversion. Oppositely, all operations in the proposed operator are performed on each decision variable, separately. So, the proposed operator is computationally efficient

[2]How to construct the restricted mating pool depends on which algorithm framework is used. In MOEA/D, the mating pool is already provided. For other frameworks like NSGA-III, it can be constructed by hand. More details are provided in Section III.

Algorithm 3 Framework of MOEA/D-EP

Input: $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_H\}$: A set of reference vectors
T: Neighborhood size
maxFEs: Maximum number of fitness evaluations allowed
S: The number of different updating rates
HL: Length of the historical memory
$g(\cdot)$: An aggregation function
Output: A set of non-dominated solutions

$N \leftarrow H, FEs \leftarrow N$
for $i = 1$ to N **do**
  $B(i) \leftarrow \{i_1, i_2, \ldots, i_T\}$, where $\boldsymbol{\lambda}_{i_1}, \boldsymbol{\lambda}_{i_2}, \ldots, \boldsymbol{\lambda}_{i_T}$ are T closest reference vectors to $\boldsymbol{\lambda}_i$
  $\boldsymbol{hm}_i \leftarrow \{\{0.5, 0.5\}, \ldots, \{0.5, 0.5\}\}, ptr_i \leftarrow 1$
end for
Initialize the population $X \leftarrow \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$
Initialize the ideal point $\boldsymbol{z}^* \leftarrow (z_1, z_2, \ldots, z_M)^T$
Initialize ***EPTable*** according to Eq. (2)
while $FEs < maxFEs$ **do**
  for $j = 1$ to N **do**
    $P \leftarrow B(j)$
    $\boldsymbol{epSet} \leftarrow \{\boldsymbol{ep}_{j,1}, \boldsymbol{ep}_{j,2}, \ldots, \boldsymbol{ep}_{j,S}\}$
    $\{\boldsymbol{u}, F, Cr\} \leftarrow Reproduction(\boldsymbol{x}_j, P, \boldsymbol{epSet}, \boldsymbol{hm}_j)$
    $FEs \leftarrow FEs + 1$
    Update $\boldsymbol{z}^*$ using $\boldsymbol{u}$
    for $i \in P$ **do**
      if $g(\boldsymbol{u}|\boldsymbol{z}^*, \boldsymbol{\lambda}_i) \leq g(\boldsymbol{x}_i|\boldsymbol{z}^*, \boldsymbol{\lambda}_i)$ **then**
        $\boldsymbol{x}_i \leftarrow \boldsymbol{u}$
        Update $\{\boldsymbol{ep}_{i,1}, \boldsymbol{ep}_{i,2}, \ldots, \boldsymbol{ep}_{i,S}\}$ using $\boldsymbol{u}$ according to Eq. (3)
        $Adaptation(\boldsymbol{hm}_i, ptr_i, HL, F, Cr)$
      end if
    end for
  end for
end while
return non-dominated solutions of $\boldsymbol{X}$

(see Section III-C for more details about the computation complexity).

III. Implementations in MOEA/D and NSGA-III

In this section, we consider two well-known MaOEA frameworks: 1) MOEA/D and 2) NSGA-III. Both of them employ a set of well-spread reference vectors, which facilitates the implementation of our proposed reproduction operator. Based on these frameworks, two enhanced algorithms, i.e., MOEA/D-EP and NSGA-III-EP, are described in Sections III-A and III-B, respectively.

A. Implementation in MOEA/D

We implement the MOEA/D-EP by replacing the SBX operator in the original MOEA/D [4] with the proposed reproduction operator. The pseudo-code of MOEA/D-EP is provided in Algorithm 3. First, we initialize the neighborhood structure, the population, the ideal point, the evolution paths, the parameter historical memories, and the corresponding pointers. Then, for each solution $\boldsymbol{x}_j$, we construct the mating pool using its neighbor solutions. Using this mating pool and solution $\boldsymbol{x}_j$, the *reproduction* procedure is called to generate an offspring $\boldsymbol{u}$. Once $\boldsymbol{u}$ is able to replace a solution $\boldsymbol{x}_i$, the associated evolution paths $\{\boldsymbol{ep}_{i,1}, \boldsymbol{ep}_{i,2}, \ldots, \boldsymbol{ep}_{i,S}\}$ are updated by $\boldsymbol{u}$. Finally, the procedure *adaptation* is called to update the corresponding historical memory.

Algorithm 4 Framework of NSGA-III-EP

Input: $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_H\}$: A set of reference vectors
N: Population size
maxFEs: Maximum number of fitness evaluations allowed
S: The number of different updating rates
HL: Length of the historical memory
Output: A set of non-dominated solutions

for $i = 1$ to H **do**
  $\boldsymbol{hm}_i \leftarrow \{\{0.5, 0.5\}, \ldots, \{0.5, 0.5\}\}, ptr_i \leftarrow 1$
end for
for $i = 1$ to N **do**
  $\pi(x_i) \leftarrow 1$
end for
Initialize the population $X \leftarrow \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$
Initialize ***EPTable*** according to Eq (2)
$FEs \leftarrow N, t \leftarrow 0$
while $FEs < maxFEs$ **do**
  $t \leftarrow t + 1, Q_t \leftarrow \{\}$
  for $i = 1$ to N **do**
    $j \leftarrow \pi(\boldsymbol{x}_i), P \leftarrow \{\boldsymbol{s}|\pi(\boldsymbol{s}) = j\}$
    $\boldsymbol{epSet} \leftarrow \{\boldsymbol{ep}_{j,1}, \boldsymbol{ep}_{j,2}, \ldots, \boldsymbol{ep}_{j,S}\}$
    $\{\boldsymbol{u}, F, Cr\} \leftarrow Reproduction(\boldsymbol{x}_i, P, \boldsymbol{epSet}, \boldsymbol{hm}_j)$
    $\boldsymbol{u}.para \leftarrow \{F, Cr\}$
    $FEs \leftarrow FEs + 1$
    $Q_t \leftarrow Q_t \cup \{\boldsymbol{u}\}$
  end for
  $(F_1, F_2, ...) \leftarrow nonDominatedSort(Q_t \cup \boldsymbol{X})$
  $i \leftarrow 1$
  repeat
    $S_t \leftarrow S_t \cup F_i, i \leftarrow i + 1$
  until $|S_t| > N$
  $F_l \leftarrow F_i, P_{t+1} = \cup_{j=1}^{l-1} F_j$
  Normalize the solutions in P_{t+1}
  for $\boldsymbol{s} \in S_t$ **do**
    $\{\pi(\boldsymbol{s}), d(\boldsymbol{s})\} \leftarrow associate(S_t, \mathbf{\Lambda})$
  end for
  for $j \leftarrow 1$ to H **do**
    $\rho_j \leftarrow \Sigma_{\boldsymbol{s} \in S_t/F_l}((\pi(\boldsymbol{s}) = j)?1{:}0)$
  end for
  $\boldsymbol{X} \leftarrow P_{t+1} \cup niching(N - |P_{t+1}|, \rho_j, \pi, d, F_l, P_{t+1})$
  for $i \in \{k|\rho_k > 0\}$ **do**
    $min\_level \leftarrow min\{k|\boldsymbol{s} \in F_k, \pi(\boldsymbol{s}) = i\}$
    $\boldsymbol{s}_0 \leftarrow \underset{\boldsymbol{s}}{argmin}\{d(\boldsymbol{s})|\pi(\boldsymbol{s}) = i, \boldsymbol{s} \in F_{min\_level}\}$
    Update $\{\boldsymbol{ep}_{i,1}, \boldsymbol{ep}_{i,2}, \ldots, \boldsymbol{ep}_{i,S}\}$ using $\boldsymbol{s}_0$ according to Eq. (3)
    if $\boldsymbol{s}_0 \in Q_t$ **then**
      $\{F, Cr\} \leftarrow \boldsymbol{s}_0.para$
      $Adaptation(\boldsymbol{hm}_i, ptr_i, HL, F, Cr)$
    end if
  end for
end while
return non-dominated solutions of $\boldsymbol{X}$

B. Implementation in NSGA-III

In this part, we discuss the implementation of NSGA-III-EP. The pseudo-code is shown in Algorithm 4. The tth generation of this algorithm is described as follows.

For each solution $\boldsymbol{x}_i$, we first find out its associated reference vector $\boldsymbol{\lambda}_j$. A mating pool P is formed using the solutions associated with $\boldsymbol{\lambda}_j$. *Reproduction* is then called to generate an offspring and the corresponding parameters. For convenience, these parameters are stored temporarily in a field named "para." After N offspring have been generated, we construct a candidate set S_t from the mixed population using

the nondominated sorting. Each solution in S_t is then associated with a reference vector based on their normalized perpendicular distance. After that, we use a niche-preservation operator to choose the remaining solutions until the size of the new population is equal to N. Finally, for each reference vector, we find out the associated solution with the minimum nondominated rank and the shortest perpendicular distance. This solution, denoted by $\boldsymbol{s}_0$, is then employed to update the corresponding evolution paths. Additionally, if $\boldsymbol{s}_0$ is generated in the current generation, the procedure *adaption* is called to update the historical memory. The procedures *nonDominatedSort*, *associate*, *niching*, and the normalization technique are introduced in the original NSGA-III algorithm, and interested readers may refer to [11] for detailed information.

The above implementation is slightly different from that in MOEA/D-EP. For instance, if a solution $\boldsymbol{x}_i$ is chosen as a parent and it is associated with $\boldsymbol{\lambda}_j$ in the last generation, $\boldsymbol{\lambda}_j$ rather than $\boldsymbol{\lambda}_i$ is selected. The reason is that $\boldsymbol{x}_i$ is close to $\boldsymbol{\lambda}_j$ in terms of perpendicular distance. As a result, it is helpful to search along the direction of $\boldsymbol{\lambda}_j$. Moreover, the set of associated solutions is chosen as the mating pool to form the difference vector in (5). This strategy is based on the fact that the number of associated solutions measures the crowdedness of the population close to the reference vector. If a reference vector associates with many solutions, there will be lots of different kinds of difference vectors formed in (5). Consequently, the diversity of the population near this reference vector is improved.

Besides, NSGA-III-EP differs from MOEA/D-EP in the updating process. For instance, when updating the evolution paths corresponding to a given reference vector, there may exist more than one associated solutions which can be used. However, it is not necessary to choose all of them since some may be dominated. Involving dominated solutions in the updating process may decrease the convergence rate. So, dominated solutions are discarded in the updating. Solutions distant to the reference vector should be eliminated too since they may change the search direction. Therefore, in NSGA-III-EP, only the solution with the minimum nondominated rank is considered. If there are more than one solution meet this requirement, the one with the smallest perpendicular distance to the reference vector is chosen.

C. Computational Complexity

For each solution, the proposed reproduction operator requires $O(S^2)$ computations to construct the Lagrange polynomial, and $O(S)$ computations to update the evolution paths and the historical memory. In NSGA-III-EP, it requires extra computations to construct the mating pool P in step 13 in Algorithm 4 and find the solution $\boldsymbol{s}_0$ in steps 35 and 36 in Algorithm 4. However, these extra computations in one generation will not exceed $O(N)$, which means it will not change the overall time complexity. Therefore, the computations of the proposed reproduction operator implemented in MOEA/D-EP and NSGA-III-EP are both $O(NS^2)$.

Note that, when constructing the Lagrange polynomial, $S-1$ is equal to the order of this polynomial which should not be too large. Thus, we set $S = O(1)$ in this paper. As a result, the time complexity of the proposed reproduction operator is equal to that of SBX or DE.

IV. Comparative Studies

In this section, numerical experiments are carried out to demonstrate the effectiveness of the proposed reproduction operator. The test problems are from the WFG test suite [50], the DTLZ test suite [51], and the Minus-DTLZ test suite [5]. First, the proposed operator is compared with other four operators (i.e., SBX [17], BLX-α [21], DE [28], and HOP [25]) in the MOEA/D framework. Then, NSGA-III-EP is compared with NSGA-III. Finally, both MOEA/D-EP and NSGA-III-EP are compared with five state-of-the-art MaOEAs (i.e., MOEA/DD [14], MOMBI2 [19], MOEA/D-DU [9], RVEA [8], and VaEA [52]). Hypervolume (HV) [53] is employed to measure both convergence and diversity of solutions obtained on the WFG and DTLZ test suite. Additionally, the averaged Hausdorff distance (Δ_p) [54] is used on the Minus-DTLZ test suite since these problems are so difficult that many algorithms cannot converge to the PF regions. Δ_p can be considered as a combination of the generational distance (GD) [55] and the inverted GD (IGD) [56]. Compared with IGD and GD, it prefers evenly spread solutions along the PF and is able to handle the outlier tradeoff. In the proposed operator, S is set to 4 and HL is set to $\lceil 0.1N \rceil$. The detailed experimental settings are provided in Section S-I in the supplementary material.

A. Comparisons With Other Reproduction Operators

1) Results on Problems With Regular PFs: Table I summaries the results of the five MOEA/D-based algorithms in handling regular PFs. As shown, MOEA/D-EP presents a clear advantage over all competitors. MOEA/D-BLX-α, MOEA/D-DE, and MOEA/D-HOP perform similarly and they are significantly outperformed by MOEA/D-EP on more than 30 test instances. MOEA/D-SBX surpasses the other competitors especially in solving three- and five-objective problems but is still inferior to MOEA/D-EP on the majority of the test instances.

Table II provides the results of comparing NSGA-III-EP with NSGA-III. The proposed operator significantly improves NSGA-III on 37 of 50 test instances. NSGA-III-EP surpasses NSGA-III on almost all the test instances on DTLZ2, DTLZ4, and WFG4–WFG9. On the contrary, NSGA-III-EP is defeated by NSGA-III on DTLZ1 and DTLZ3.

According to the above results, we can conclude that the proposed reproduction operator improves the performance on problems with regular PFs. This can be ascribed to the fact that employing the well-spread reference vectors in the reproduction operator helps to maintain the convergence and diversity of the offspring solutions.

To visually understand this conclusion, Fig. 6 plots the final solutions obtained by the five MOEA/D-based algorithms on

TABLE I
MEDIAN AND IQR RESULTS ON PROBLEMS WITH REGULAR PFS OBTAINED BY MOEA/D EQUIPPED WITH FIVE DIFFERENT OPERATORS. THE HV INDICATOR IS USED FOR ALL TEST INSTANCES. THE BEST AND THE SECOND BEST RESULTS FOR EACH TEST INSTANCE ARE SHOWN WITH DARK AND LIGHT GRAY BACKGROUND, RESPECTIVELY

Problem	M	MOEA/D-EP	MOEA/D-SBX	MOEA/D-BLX-α	MOEA/D-DE	MOEA/D-HOP
WFG4	3	$7.41e-1(4.0e-3)$	$7.60e-1(4.0e-3)$ ○	$7.12e-1(5.0e-3)$ ●	$6.83e-1(9.5e-3)$ ●	$6.77e-1(1.1e-2)$ ●
	5	$8.52e-1(1.2e-2)$	$8.55e-1(9.1e-3)$ ⋆	$8.10e-1(1.3e-2)$ ●	$7.58e-1(8.7e-3)$ ●	$7.57e-1(1.3e-2)$ ●
	8	$6.93e-1(2.3e-2)$	$6.96e-1(3.1e-2)$ ⋆	$6.62e-1(4.2e-2)$ ●	$6.26e-1(2.7e-2)$ ●	$6.18e-1(1.4e-2)$ ●
	10	$7.40e-1(3.3e-2)$	$7.09e-1(3.9e-2)$ ●	$7.15e-1(4.4e-2)$ ⋆	$6.40e-1(3.1e-2)$ ●	$6.50e-1(3.0e-2)$ ●
	15	$6.20e-1(6.3e-2)$	$5.34e-1(9.7e-2)$ ●	$5.97e-1(7.5e-2)$ ●	$6.22e-1(1.1e-2)$ ●	$6.28e-1(1.5e-2)$ ⋆
WFG5	3	$7.46e-1(8.6e-4)$	$7.40e-1(1.7e-3)$ ●	$6.90e-1(8.4e-3)$ ●	$7.29e-1(5.3e-3)$ ●	$7.27e-1(6.9e-3)$ ●
	5	$8.51e-1(2.8e-3)$	$8.46e-1(9.0e-3)$ ●	$8.24e-1(3.7e-3)$ ●	$8.28e-1(6.8e-3)$ ●	$8.26e-1(7.8e-3)$ ●
	8	$7.27e-1(2.4e-2)$	$7.19e-1(2.1e-2)$ ⋆	$7.02e-1(2.2e-2)$ ●	$7.20e-1(2.6e-2)$ ⋆	$7.16e-1(2.8e-2)$ ⋆
	10	$7.59e-1(2.1e-2)$	$7.47e-1(1.2e-2)$ ⋆	$7.19e-1(1.7e-2)$ ●	$7.48e-1(2.0e-2)$ ●	$7.48e-1(2.1e-2)$ ⋆
	15	$6.04e-1(2.7e-2)$	$5.97e-1(2.0e-2)$ ⋆	$5.64e-1(3.6e-2)$ ●	$6.52e-1(4.0e-2)$ ○	$6.50e-1(3.9e-2)$ ○
WFG6	3	$7.31e-1(1.5e-3)$	$7.37e-1(8.0e-3)$ ○	$6.95e-1(9.0e-3)$ ●	$7.30e-1(2.9e-3)$ ●	$7.27e-1(2.4e-3)$ ●
	5	$8.35e-1(7.7e-3)$	$7.75e-1(4.9e-2)$ ●	$8.47e-1(1.2e-2)$ ○	$8.06e-1(1.4e-2)$ ●	$8.17e-1(9.0e-3)$ ●
	8	$6.23e-1(5.9e-2)$	$5.76e-1(6.4e-2)$ ●	$7.17e-1(7.2e-2)$ ○	$6.21e-1(2.9e-2)$ ⋆	$6.95e-1(2.6e-2)$ ○
	10	$6.88e-1(3.9e-2)$	$5.91e-1(4.8e-2)$ ●	$7.19e-1(8.3e-2)$ ⋆	$6.09e-1(4.8e-2)$ ●	$7.43e-1(1.4e-2)$ ○
	15	$5.34e-1(7.2e-2)$	$4.13e-1(5.2e-2)$ ●	$4.19e-1(5.4e-2)$ ●	$5.24e-1(7.6e-2)$ ⋆	$5.36e-1(6.1e-2)$ ⋆
WFG7	3	$7.27e-1(7.4e-3)$	$7.47e-1(1.6e-2)$ ○	$7.41e-1(7.4e-3)$ ○	$6.45e-1(9.6e-3)$ ●	$6.43e-1(7.4e-3)$ ●
	5	$8.49e-1(1.3e-2)$	$8.46e-1(5.3e-3)$ ⋆	$8.72e-1(1.0e-2)$ ○	$7.06e-1(1.7e-2)$ ●	$7.01e-1(1.4e-2)$ ●
	8	$6.51e-1(4.3e-2)$	$6.43e-1(3.1e-2)$ ⋆	$7.29e-1(6.9e-2)$ ○	$5.52e-1(3.7e-2)$ ●	$5.76e-1(3.3e-2)$ ●
	10	$7.18e-1(3.0e-2)$	$6.57e-1(1.4e-2)$ ●	$7.35e-1(3.4e-2)$ ⋆	$6.30e-1(1.6e-2)$ ●	$6.35e-1(3.3e-2)$ ●
	15	$6.07e-1(1.6e-3)$	$4.40e-1(4.2e-3)$ ●	$5.89e-1(4.3e-2)$ ●	$5.50e-1(7.1e-2)$ ●	$5.98e-1(2.5e-2)$ ⋆
WFG8	3	$6.84e-1(9.0e-3)$	$7.14e-1(5.6e-3)$ ○	$6.68e-1(7.9e-3)$ ●	$5.74e-1(1.7e-2)$ ●	$5.80e-1(1.4e-2)$ ●
	5	$7.46e-1(4.7e-2)$	$7.33e-1(7.0e-2)$ ⋆	$7.15e-1(2.6e-2)$ ●	$6.16e-1(1.5e-2)$ ●	$6.19e-1(1.2e-2)$ ●
	8	$4.60e-1(5.7e-2)$	$3.37e-1(4.3e-2)$ ●	$3.55e-1(3.3e-2)$ ●	$3.34e-1(3.1e-2)$ ●	$3.62e-1(2.0e-2)$ ●
	10	$5.05e-1(6.3e-2)$	$3.51e-1(5.0e-2)$ ●	$3.61e-1(4.5e-2)$ ●	$3.65e-1(4.8e-2)$ ●	$4.11e-1(5.2e-2)$ ●
	15	$4.16e-1(9.2e-2)$	$4.39e-1(2.9e-1)$ ⋆	$2.36e-1(5.1e-2)$ ●	$3.36e-1(6.6e-2)$ ●	$3.58e-1(6.3e-2)$ ●
WFG9	3	$7.08e-1(1.3e-3)$	$6.84e-1(1.3e-2)$ ●	$6.91e-1(7.6e-3)$ ●	$6.95e-1(9.2e-3)$ ●	$6.97e-1(8.1e-3)$ ●
	5	$7.97e-1(1.7e-3)$	$7.72e-1(6.1e-2)$ ⋆	$7.82e-1(1.8e-2)$ ●	$7.73e-1(8.7e-3)$ ●	$7.70e-1(1.3e-2)$ ●
	8	$6.19e-1(1.4e-2)$	$5.25e-1(1.3e-1)$ ●	$5.97e-1(5.7e-2)$ ⋆	$6.07e-1(6.4e-2)$ ⋆	$6.07e-1(4.4e-2)$ ⋆
	10	$5.59e-1(7.3e-2)$	$5.06e-1(9.1e-2)$ ●	$5.38e-1(6.7e-2)$ ⋆	$5.78e-1(3.5e-2)$ ⋆	$5.77e-1(4.6e-2)$ ⋆
	15	$4.84e-1(1.5e-1)$	$3.42e-1(1.0e-1)$ ●	$3.40e-1(8.7e-2)$ ●	$4.71e-1(1.3e-1)$ ⋆	$4.30e-1(8.9e-2)$ ⋆
DTLZ1	3	$9.62e-1(1.0e-1)$	$9.94e-1(6.7e-5)$ ○	$6.16e-1(4.2e-1)$ ●	$9.80e-1(4.1e-2)$ ⋆	$9.53e-1(1.3e-1)$ ⋆
	5	$1.00e+0(1.3e-5)$	$1.00e+0(8.1e-6)$ ⋆	$8.08e-1(2.6e-1)$ ●	$1.00e+0(1.4e-5)$ ●	$1.00e+0(9.2e-6)$ ●
	8	$1.00e+0(6.9e-7)$	$1.00e+0(2.9e-5)$ ●	$9.69e-1(7.3e-2)$ ●	$1.00e+0(1.1e-6)$ ⋆	$1.00e+0(1.3e-6)$ ●
	10	$1.00e+0(0.0e+0)$	$1.00e+0(5.9e-6)$ ●	$9.94e-1(2.7e-2)$ ⋆	$1.00e+0(0.0e+0)$ ⋆	$1.00e+0(2.2e-7)$ ⋆
	15	$1.00e+0(0.0e+0)$	$1.00e+0(1.5e-4)$ ●	$1.00e+0(0.0e+0)$ ⋆	$1.00e+0(0.0e+0)$ ⋆	$1.00e+0(0.0e+0)$ ⋆
DTLZ2	3	$9.43e-1(2.0e-4)$	$9.45e-1(1.8e-5)$ ○	$9.40e-1(7.7e-4)$ ●	$9.35e-1(1.9e-3)$ ●	$9.35e-1(1.6e-3)$ ●
	5	$9.93e-1(2.0e-4)$	$9.94e-1(6.9e-5)$ ○	$9.93e-1(2.1e-4)$ ⋆	$9.89e-1(5.0e-4)$ ●	$9.89e-1(5.9e-4)$ ●
	8	$1.00e+0(1.5e-5)$	$1.00e+0(1.9e-5)$ ⋆	$1.00e+0(2.7e-5)$ ●	$9.98e-1(5.8e-4)$ ●	$9.98e-1(4.7e-4)$ ●
	10	$1.00e+0(6.8e-6)$	$1.00e+0(5.5e-6)$ ●	$1.00e+0(6.5e-6)$ ⋆	$1.00e+0(3.2e-5)$ ●	$1.00e+0(2.8e-5)$ ●
	15	$1.00e+0(5.9e-7)$	$1.00e+0(7.4e-7)$ ●	$1.00e+0(5.9e-7)$ ⋆	$1.00e+0(3.5e-5)$ ●	$1.00e+0(8.8e-6)$ ●
DTLZ3	3	$3.64e-1(4.3e-1)$	$9.44e-1(4.9e-4)$ ○	$7.72e-3(3.5e-2)$ ●	$6.68e-1(4.1e-1)$ ⋆	$5.26e-1(4.5e-1)$ ⋆
	5	$2.69e-1(4.0e-1)$	$9.94e-1(1.0e-4)$ ○	$0.00e+0(0.0e+0)$ ●	$8.21e-1(3.6e-1)$ ○	$8.88e-1(3.0e-1)$ ○
	8	$5.56e-1(4.8e-1)$	$8.89e-1(1.8e-1)$ ○	$4.22e-2(1.9e-1)$ ●	$9.48e-1(2.2e-1)$ ○	$9.93e-1(1.3e-2)$ ○
	10	$9.98e-1(8.3e-3)$	$1.00e+0(2.3e-5)$ ⋆	$5.00e-2(2.2e-1)$ ●	$1.00e+0(5.1e-5)$ ●	$1.00e+0(8.0e-5)$ ●
	15	$9.57e-1(1.3e-1)$	$7.30e-1(2.3e-1)$ ●	$9.27e-1(2.4e-1)$ ⋆	$1.00e+0(6.0e-6)$ ●	$1.00e+0(1.0e-5)$ ●
DTLZ4	3	$9.45e-1(2.5e-5)$	$8.23e-1(1.7e-1)$ ⋆	$9.42e-1(6.3e-4)$ ●	$9.39e-1(2.0e-3)$ ●	$9.39e-1(1.5e-3)$ ●
	5	$9.94e-1(6.8e-5)$	$9.85e-1(1.1e-2)$ ●	$9.93e-1(2.6e-3)$ ●	$9.93e-1(3.3e-4)$ ●	$9.93e-1(3.2e-4)$ ●
	8	$1.00e+0(1.8e-5)$	$9.97e-1(2.5e-3)$ ●	$9.99e-1(2.4e-4)$ ●	$1.00e+0(1.9e-5)$ ●	$1.00e+0(2.6e-5)$ ●
	10	$1.00e+0(4.6e-6)$	$9.99e-1(1.6e-3)$ ●	$1.00e+0(2.1e-5)$ ⋆	$1.00e+0(6.3e-6)$ ●	$1.00e+0(7.5e-6)$ ●
	15	$1.00e+0(3.7e-7)$	$1.00e+0(1.8e-4)$ ●	$1.00e+0(4.4e-6)$ ●	$1.00e+0(6.9e-7)$ ●	$1.00e+0(6.0e-7)$ ●
● / ⋆ / ○			26 / 14 / 10	33 / 12 / 5	36 / 11 / 3	33 / 12 / 5

"●" indicates that the enhanced algorithm significantly outperforms the peer algorithm at a 0.05 significance level by the Wilcoxon signed rank test, whereas "○" indicates the opposite. If no significant difference is detected, it will be marked by the symbol "⋆". They have the same meanings in other tables.

the 15-objective DTLZ1 problem. These solutions are corresponding to the run that obtains the closest result to the median HV value. All the four competitors fail to find out the boundary solutions on some objectives. Moreover, both MOEA/D-DE and MOEA/D-HOP obtain solutions located outside the PF. On the contrary, MOEA/D-EP shows best performance in terms of both convergence and coverage. For NSGA-III-EP and NSGA-III, Figs. 7 and 8 plot the final solutions obtained on the 15-objective WFG5 and WFG8 problems, respectively. On all these test instances, both two algorithms reach the PFs. However, NSGA-III shows worse performance in terms of coverage. For example, it fails to cover the 6th to the 13th objectives on WFG5 and the first four objectives on WFG8. Whereas, NSGA-III-EP covers all the objectives on these test instances.

2) Results on Problems With Irregular PFs: Table III provides the results obtained by the five MOEA/D-based algorithms in handling irregular PFs. MOEA/D-EP obtains the best and the second best results on 18 and 11 test instances, respectively. It shows significant advantages on DTLZ1^{-1}, DTLZ2^{-1}, DTLZ4^{-1}, and WFG1–WFG3. MOEA/D-SBX performs relatively poor and is surpassed by MOEA/D-EP on 36 test instances, while MOEA/D-BLX-α performs well and obtains the best results on ten test instances. MOEA/D-DE and MOEA/D-HOP show similar performance on most test instances. HOP slightly strengthens the DE operator by hybridizing with a nonlinear mutation. However, its advantages over DE can only be observed in high-dimensional objective space. When the number of objectives is small, it seems that the hybrid scheme in HOP even deteriorates the performance of the canonical DE operator.

Table IV presents the comparison between NSGA-III-EP and NSGA-III. It is found that the proposed operator strengthens NSGA-III on WFG2, DTLZ5, DTLZ6, and

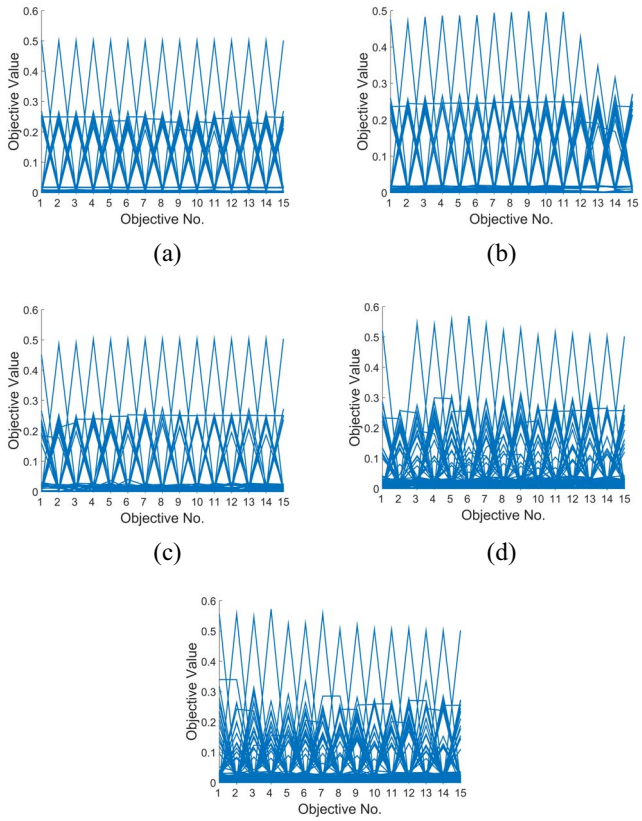


Fig. 6. Final solution sets on the 15-objective DTLZ1, shown by parallel coordinates. (a) MOEA/D-EP. (b) MOEA/D-SBX. (c) MOEA/D-BLX-α. (d) MOEA/D-DE. (e) MOEA/D-HOP.

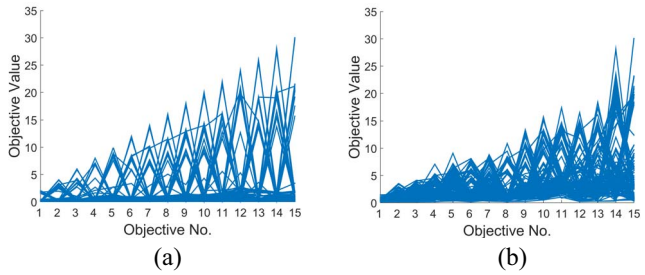


Fig. 7. Final solution sets on the 15-objective WFG5, shown by parallel coordinates. (a) NSGA-III-EP. (b) NSGA-III.

DTLZ1^{-1}–DTLZ4^{-1}, but deteriorates the performance on WFG1 and WFG3. On DTLZ7 which has discontinuous PFs, the proposed operator does not perform well on 3-, 5-, and 8-objective test instances. Nevertheless, it is superior to SBX in high-dimensional objective space.

Clearly, the above experimental results demonstrate that the proposed operator is competent to handle irregular PFs. However, its effectiveness is also related to the PF shapes when considering certain problems. Some conclusions are summarized as follows.

1) On problems with rotated or convex PFs (i.e., Minus-DTLZ problems), the proposed operator significantly strengthens the algorithm performance.

TABLE II
MEDIAN AND IQR RESULTS ON PROBLEMS WITH REGULAR PFS OBTAINED BY NSGA-III-EP AND NSGA-III. THE HV INDICATOR IS USED FOR ALL TEST INSTANCES. THE BETTER RESULT FOR EACH TEST INSTANCE IS SHOWN WITH DARK GRAY BACKGROUND

Problem	M	NSGA-III-EP	NSGA-III
WFG4	3	$7.58e-1(4.5e-3)$	$7.49e-1(7.6e-3)$ $\bullet$
	5	$8.86e-1(4.5e-3)$	$7.74e-1(1.1e-2)$ $\bullet$
	8	$9.47e-1(2.2e-3)$	$7.97e-1(1.5e-2)$ $\bullet$
	10	$9.71e-1(1.2e-3)$	$8.47e-1(1.5e-2)$ $\bullet$
	15	$9.79e-1(2.4e-3)$	$4.31e-1(1.7e-2)$ $\bullet$
WFG5	3	$7.50e-1(1.3e-3)$	$7.25e-1(6.6e-3)$ $\bullet$
	5	$8.72e-1(1.5e-3)$	$7.75e-1(7.4e-3)$ $\bullet$
	8	$8.97e-1(1.1e-2)$	$8.00e-1(1.9e-2)$ $\bullet$
	10	$9.15e-1(9.8e-3)$	$8.49e-1(1.3e-2)$ $\bullet$
	15	$9.11e-1(9.3e-3)$	$5.03e-1(2.4e-2)$ $\bullet$
WFG6	3	$7.40e-1(2.4e-4)$	$7.30e-1(6.7e-3)$ $\bullet$
	5	$8.56e-1(1.9e-4)$	$7.44e-1(9.2e-3)$ $\bullet$
	8	$8.87e-1(4.0e-4)$	$7.67e-1(5.1e-2)$ $\bullet$
	10	$8.94e-1(6.2e-4)$	$8.51e-1(4.7e-2)$ $\bullet$
	15	$8.86e-1(1.7e-3)$	$5.49e-1(4.3e-2)$ $\bullet$
WFG7	3	$7.34e-1(6.8e-3)$	$7.54e-1(6.1e-3)$ $\circ$
	5	$8.85e-1(5.5e-3)$	$7.84e-1(1.1e-2)$ $\bullet$
	8	$9.56e-1(4.6e-3)$	$8.17e-1(3.6e-2)$ $\bullet$
	10	$9.82e-1(1.7e-3)$	$9.09e-1(1.2e-2)$ $\bullet$
	15	$9.92e-1(3.4e-3)$	$4.98e-1(3.1e-2)$ $\bullet$
WFG8	3	$6.91e-1(5.3e-3)$	$6.96e-1(4.8e-3)$ $\circ$
	5	$8.16e-1(7.4e-3)$	$6.67e-1(2.9e-2)$ $\bullet$
	8	$8.71e-1(1.2e-2)$	$7.26e-1(5.9e-2)$ $\bullet$
	10	$9.14e-1(1.0e-2)$	$7.06e-1(2.7e-2)$ $\bullet$
	15	$8.84e-1(2.6e-2)$	$4.09e-1(1.8e-1)$ $\bullet$
WFG9	3	$7.12e-1(1.2e-3)$	$7.11e-1(1.3e-2)$ $\star$
	5	$8.05e-1(1.9e-3)$	$7.52e-1(4.9e-3)$ $\bullet$
	8	$8.01e-1(1.1e-2)$	$7.47e-1(1.0e-2)$ $\bullet$
	10	$8.09e-1(7.0e-3)$	$7.81e-1(1.4e-2)$ $\bullet$
	15	$7.68e-1(1.1e-2)$	$5.87e-1(2.5e-2)$ $\bullet$
DTLZ1	3	$8.71e-1(1.6e-1)$	$9.90e-1(1.8e-3)$ $\circ$
	5	$9.90e-1(1.3e-2)$	$9.98e-1(5.2e-4)$ $\star$
	8	$2.10e-1(3.0e-1)$	$9.99e-1(3.9e-4)$ $\circ$
	10	$4.47e-1(3.3e-1)$	$9.99e-1(2.0e-4)$ $\circ$
	15	$6.51e-2(1.7e-1)$	$9.79e-1(5.2e-2)$ $\circ$
DTLZ2	3	$9.44e-1(6.5e-5)$	$9.37e-1(1.9e-3)$ $\bullet$
	5	$9.94e-1(6.4e-5)$	$9.88e-1(1.1e-3)$ $\bullet$
	8	$1.00e+0(2.0e-5)$	$9.93e-1(1.9e-3)$ $\bullet$
	10	$1.00e+0(5.4e-6)$	$9.97e-1(9.2e-4)$ $\bullet$
	15	$1.00e+0(4.4e-7)$	$9.96e-1(2.3e-3)$ $\bullet$
DTLZ3	3	$4.42e-1(3.6e-1)$	$9.35e-1(3.2e-3)$ $\circ$
	5	$3.53e-1(3.8e-1)$	$9.87e-1(1.3e-3)$ $\circ$
	8	$0.00e+0(0.0e+0)$	$9.95e-1(3.1e-3)$ $\circ$
	10	$0.00e+0(0.0e+0)$	$9.99e-1(1.7e-3)$ $\circ$
	15	$0.00e+0(0.0e+0)$	$9.00e-1(3.1e-1)$ $\circ$
DTLZ4	3	$9.45e-1(2.6e-5)$	$9.35e-1(2.4e-2)$ $\bullet$
	5	$9.94e-1(8.8e-5)$	$9.92e-1(4.5e-4)$ $\bullet$
	8	$1.00e+0(1.7e-5)$	$9.98e-1(5.3e-4)$ $\bullet$
	10	$1.00e+0(4.6e-6)$	$9.99e-1(1.7e-4)$ $\bullet$
	15	$1.00e+0(5.7e-7)$	$1.00e+0(2.9e-4)$ $\bullet$
$\bullet$ / $\star$ / $\circ$			37 / 2 / 11

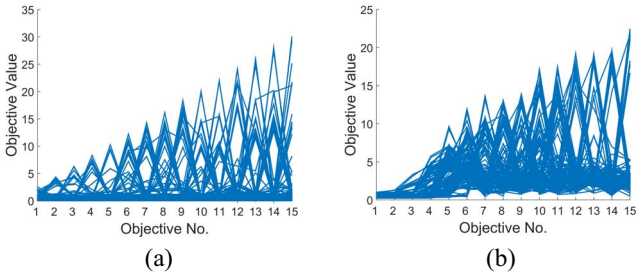


Fig. 8. Final solution sets on the 15-objective WFG8, shown by parallel coordinates. (a) NSGA-III-EP. (b) NSGA-III.

Take three-objective DTLZ1^{-1} as an example (shown in Fig. 9), NSGA-III is unable to preserve the population diversity since most solutions are located near the boundaries. Oppositely, the solutions obtained by

TABLE III
MEDIAN AND IQR RESULTS ON PROBLEMS WITH IRREGULAR PFS OBTAINED BY MOEA/D EQUIPPED WITH FIVE DIFFERENT OPERATORS. THE HV INDICATOR IS USED FOR WFG AND DTLZ PROBLEMS AND THE Δ_p INDICATOR IS USED FOR MINUS-DTLZ PROBLEMS. THE BEST AND THE SECOND BEST RESULTS FOR EACH TEST INSTANCE ARE SHOWN WITH DARK AND LIGHT GRAY BACKGROUND, RESPECTIVELY

Problem	M	MOEA/D-EP	MOEA/D-SBX	MOEA/D-BLX-α	MOEA/D-DE	MOEA/D-HOP
WFG1	3	$4.93e-1(8.6e-3)$	$5.84e-1(2.7e-2)$ ○	$4.97e-1(1.1e-2)$ ⋆	$4.69e-1(5.1e-3)$ ●	$4.71e-1(4.5e-3)$ ●
	5	$7.27e-1(3.9e-2)$	$8.23e-1(3.8e-2)$ ○	$6.12e-1(6.7e-2)$ ●	$4.22e-1(1.0e-2)$ ●	$4.44e-1(3.0e-2)$ ●
	8	$9.31e-1(5.4e-3)$	$8.03e-1(7.7e-2)$ ●	$8.43e-1(4.0e-2)$ ●	$6.29e-1(5.4e-2)$ ●	$7.06e-1(5.3e-2)$ ●
	10	$9.39e-1(2.2e-3)$	$7.12e-1(6.0e-2)$ ●	$8.35e-1(5.2e-2)$ ●	$7.22e-1(3.5e-2)$ ●	$8.84e-1(4.0e-2)$ ●
	15	$8.86e-1(4.3e-2)$	$4.54e-1(2.9e-2)$ ●	$7.58e-1(8.6e-2)$ ●	$8.61e-1(4.5e-2)$ ●	$9.09e-1(1.5e-2)$ ⋆
WFG2	3	$8.90e-1(2.0e-2)$	$7.84e-1(2.8e-2)$ ●	$8.54e-1(5.9e-2)$ ●	$8.38e-1(4.6e-2)$ ●	$8.63e-1(1.3e-2)$ ●
	5	$9.66e-1(4.0e-3)$	$8.22e-1(5.5e-2)$ ●	$8.55e-1(6.8e-2)$ ●	$9.50e-1(4.3e-3)$ ●	$9.44e-1(5.2e-3)$ ●
	8	$9.58e-1(2.9e-3)$	$8.09e-1(6.9e-2)$ ●	$8.52e-1(7.1e-2)$ ●	$9.05e-1(5.6e-2)$ ●	$9.21e-1(3.8e-2)$ ●
	10	$9.63e-1(2.6e-3)$	$8.25e-1(5.8e-2)$ ●	$8.74e-1(7.8e-2)$ ●	$9.42e-1(2.9e-2)$ ●	$9.48e-1(3.5e-3)$ ●
	15	$9.59e-1(3.7e-3)$	$7.96e-1(3.5e-2)$ ●	$8.73e-1(7.8e-2)$ ●	$9.11e-1(5.7e-2)$ ●	$9.43e-1(3.4e-2)$ ●
WFG3	3	$7.23e-1(7.7e-3)$	$6.94e-1(2.7e-2)$ ●	$7.30e-1(1.0e-2)$ ○	$6.87e-1(1.9e-2)$ ●	$6.85e-1(1.1e-2)$ ●
	5	$6.48e-1(1.0e-2)$	$6.22e-1(1.7e-2)$ ●	$6.25e-1(1.1e-2)$ ●	$6.03e-1(1.3e-2)$ ●	$6.07e-1(1.8e-2)$ ●
	8	$5.02e-1(1.5e-2)$	$4.26e-1(2.7e-2)$ ●	$3.85e-1(2.3e-2)$ ●	$4.92e-1(1.5e-2)$ ⋆	$4.96e-1(2.6e-2)$ ⋆
	10	$3.36e-1(4.6e-2)$	$3.21e-1(3.1e-2)$ ●	$3.05e-1(2.1e-3)$ ⋆	$4.36e-1(2.3e-2)$ ○	$4.40e-1(2.3e-2)$ ○
	15	$2.47e-1(1.3e-3)$	$2.43e-1(1.8e-3)$ ●	$2.43e-1(3.0e-3)$ ●	$2.44e-1(1.6e-3)$ ●	$2.45e-1(2.1e-3)$ ●
DTLZ5	3	$1.83e-1(3.7e-4)$	$1.82e-1(9.1e-5)$ ●	$1.83e-1(2.0e-4)$ ●	$1.83e-1(3.2e-4)$ ○	$1.83e-1(4.4e-4)$ ⋆
	5	$1.27e-1(3.6e-4)$	$1.27e-1(4.3e-4)$ ⋆	$1.27e-1(3.1e-4)$ ⋆	$1.27e-1(3.2e-4)$ ⋆	$1.27e-1(3.0e-4)$ ⋆
	8	$1.04e-1(2.8e-4)$	$1.04e-1(2.9e-4)$ ⋆	$1.04e-1(3.5e-4)$ ○	$1.04e-1(3.0e-4)$ ○	$1.04e-1(2.1e-4)$ ⋆
	10	$1.00e-1(3.4e-4)$	$1.00e-1(2.6e-4)$ ⋆	$1.00e-1(2.3e-4)$ ⋆	$1.00e-1(3.3e-4)$ ⋆	$1.00e-1(2.7e-4)$ ⋆
	15	$9.44e-2(3.6e-4)$	$9.46e-2(3.5e-4)$ ⋆	$9.45e-2(2.8e-4)$ ⋆	$9.42e-2(3.1e-4)$ ⋆	$9.43e-2(3.7e-4)$ ⋆
DTLZ6	3	$1.82e-1(6.4e-6)$	$0.00e+0(0.0e+0)$ ●	$1.82e-1(5.4e-5)$ ○	$1.82e-1(4.1e-5)$ ○	$1.82e-1(1.5e-5)$ ○
	5	$1.27e-1(3.5e-4)$	$6.03e-2(2.3e-2)$ ●	$1.27e-1(3.1e-4)$ ⋆	$1.27e-1(2.5e-4)$ ⋆	$1.27e-1(3.6e-4)$ ⋆
	8	$1.04e-1(3.1e-4)$	$1.56e-2(1.3e-2)$ ●	$1.04e-1(2.4e-4)$ ⋆	$1.04e-1(2.7e-4)$ ⋆	$1.04e-1(3.3e-4)$ ⋆
	10	$1.00e-1(3.2e-4)$	$2.68e-2(2.5e-2)$ ●	$1.00e-1(2.9e-4)$ ⋆	$1.00e-1(2.5e-4)$ ⋆	$1.00e-1(3.6e-4)$ ⋆
	15	$9.44e-2(2.8e-4)$	$1.50e-2(1.7e-2)$ ●	$9.44e-2(3.1e-4)$ ⋆	$9.43e-2(2.7e-4)$ ⋆	$9.44e-2(2.8e-4)$ ⋆
DTLZ7	3	$2.50e-1(1.6e-2)$	$2.54e-1(1.2e-2)$ ○	$2.55e-1(9.9e-4)$ ⋆	$2.41e-1(1.0e-2)$ ●	$2.45e-1(4.1e-3)$ ●
	5	$1.42e-1(1.1e-3)$	$1.44e-1(8.5e-4)$ ○	$1.45e-1(2.7e-3)$ ○	$5.16e-2(2.3e-2)$ ●	$5.69e-2(2.1e-2)$ ●
	8	$3.62e-5(7.9e-6)$	$3.33e-3(1.4e-2)$ ⋆	$6.28e-5(1.3e-4)$ ⋆	$2.20e-3(9.7e-3)$ ●	$3.35e-5(4.8e-6)$ ⋆
	10	$2.29e-6(7.1e-7)$	$7.08e-5(2.0e-4)$ ⋆	$2.38e-5(1.0e-4)$ ●	$1.46e-5(5.6e-5)$ ⋆	$8.71e-6(2.9e-5)$ ⋆
	15	$3.18e-6(7.3e-6)$	$1.39e-5(3.4e-5)$ ⋆	$5.62e-6(7.0e-6)$ ⋆	$2.41e-5(9.0e-5)$ ⋆	$4.84e-6(1.2e-5)$ ⋆
DTLZ1^{-1}	3	$4.34e+1(2.0e+0)$	$4.06e+1(1.7e+0)$ ○	$5.22e+1(5.2e+0)$ ●	$6.06e+1(4.8e+0)$ ●	$5.91e+1(4.3e+0)$ ●
	5	$6.90e+1(2.1e-1)$	$7.22e+1(2.4e+0)$ ●	$6.71e+1(3.9e-1)$ ○	$6.84e+1(1.6e+0)$ ○	$6.77e+1(5.8e-1)$ ○
	8	$2.44e+2(5.8e+0)$	$2.63e+2(1.5e+1)$ ●	$2.61e+2(1.7e+1)$ ●	$2.61e+2(1.5e+1)$ ●	$2.57e+2(1.8e+1)$ ●
	10	$2.56e+2(6.2e+0)$	$2.64e+2(8.7e+0)$ ●	$2.59e+2(1.1e+1)$ ⋆	$2.73e+2(1.0e+1)$ ●	$2.59e+2(6.3e+0)$ ⋆
	15	$3.22e+2(7.2e+0)$	$3.36e+2(1.1e+1)$ ●	$3.38e+2(1.4e+1)$ ●	$3.29e+2(1.4e+1)$ ⋆	$3.16e+2(2.0e+1)$ ○
DTLZ2^{-1}	3	$2.52e-1(5.0e-3)$	$2.58e-1(7.2e-3)$ ●	$2.57e-1(4.1e-3)$ ●	$2.59e-1(5.0e-3)$ ●	$2.60e-1(7.2e-3)$ ●
	5	$8.12e-1(1.5e-3)$	$8.14e-1(3.1e-3)$ ●	$8.16e-1(2.8e-3)$ ●	$8.14e-1(5.1e-3)$ ⋆	$8.18e-1(4.5e-3)$ ●
	8	$2.12e+0(3.5e-3)$	$2.12e+0(5.4e-3)$ ●	$2.10e+0(4.8e-3)$ ○	$2.14e+0(1.5e-2)$ ●	$2.15e+0(1.2e-2)$ ●
	10	$2.42e+0(1.5e-2)$	$2.39e+0(9.6e-3)$ ○	$2.39e+0(1.0e-2)$ ○	$2.37e+0(9.0e-3)$ ○	$2.42e+0(9.6e-3)$ ⋆
	15	$3.05e+0(2.0e-2)$	$3.12e+0(2.0e-2)$ ●	$3.07e+0(1.5e-2)$ ●	$3.06e+0(1.9e-2)$ ⋆	$3.05e+0(1.2e-2)$ ⋆
DTLZ3^{-1}	3	$2.17e+3(9.3e+0)$	$2.20e+3(9.5e-1)$ ●	$2.11e+3(1.6e+1)$ ○	$2.07e+3(1.6e+1)$ ○	$2.06e+3(1.9e+1)$ ○
	5	$2.19e+3(5.6e+0)$	$2.20e+3(1.7e-1)$ ●	$2.09e+3(1.6e+1)$ ○	$2.07e+3(1.7e+1)$ ○	$2.07e+3(2.9e+1)$ ○
	8	$2.16e+3(1.9e+1)$	$2.20e+3(2.2e+0)$ ●	$2.08e+3(2.6e+1)$ ○	$2.05e+3(3.0e+1)$ ○	$2.05e+3(3.2e+1)$ ○
	10	$2.19e+3(2.4e+0)$	$2.20e+3(2.0e-1)$ ●	$2.09e+3(2.8e+1)$ ○	$2.09e+3(3.8e+1)$ ○	$2.09e+3(3.2e+1)$ ○
	15	$2.20e+3(4.7e-1)$	$2.20e+3(2.8e-1)$ ●	$2.20e+3(1.2e+0)$ ○	$2.20e+3(4.8e+0)$ ○	$2.20e+3(7.3e+0)$ ○
DTLZ4^{-1}	3	$2.54e-1(6.5e-3)$	$5.83e-1(6.9e-1)$ ⋆	$2.63e-1(9.1e-3)$ ●	$2.85e-1(2.2e-2)$ ●	$3.20e-1(1.3e-1)$ ●
	5	$8.16e-1(1.8e-3)$	$1.19e+0(4.5e-1)$ ●	$8.16e-1(2.4e-3)$ ⋆	$8.38e-1(4.7e-2)$ ●	$8.23e-1(1.3e-2)$ ●
	8	$2.13e+0(7.4e-3)$	$2.37e+0(1.4e-1)$ ●	$2.21e+0(1.2e-1)$ ⋆	$2.08e+0(8.2e-2)$ ○	$2.15e+0(1.1e-1)$ ⋆
	10	$2.48e+0(2.1e-2)$	$2.60e+0(1.4e-1)$ ●	$2.50e+0(6.6e-2)$ ⋆	$2.56e+0(1.1e-1)$ ●	$2.51e+0(1.0e-1)$ ⋆
	15	$3.06e+0(8.2e-3)$	$3.29e+0(1.1e-1)$ ●	$3.13e+0(8.1e-2)$ ●	$3.21e+0(7.1e-2)$ ●	$3.15e+0(6.8e-2)$ ●
● / ⋆ / ○			36 / 8 / 6	22 / 16 / 12	25 / 13 / 12	22 / 19 / 9

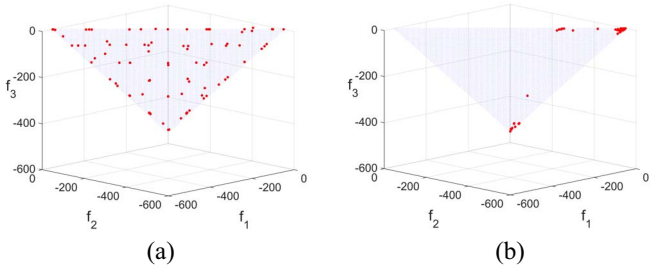


Fig. 9. Final solution sets on the three-objective DTLZ1^{-1}. (a) NSGA-III-EP. (b) NSGA-III.

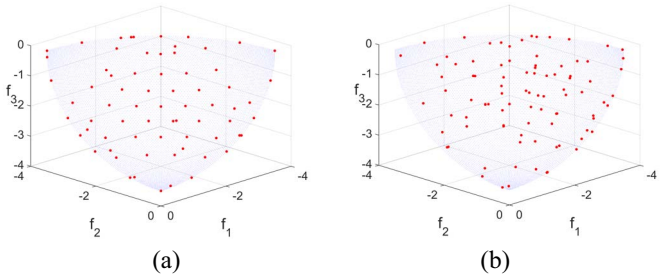


Fig. 10. Final solution sets on the three-objective DTLZ4^{-1}. (a) NSGA-III-EP. (b) NSGA-III.

NSGA-III-EP are spread over the whole PF. This is also observed on three-objective DTLZ4^{-1} as shown in Fig. 10, where NSGA-III-EP has a better distribution uniformity than NSGA-III.

2) On problems with discontinuous PFs (i.e., WFG2 and DTLZ7), the proposed operator decreases the performance in low-dimensional objective space but increases the performance in high-dimensional objective space. As shown in Fig. 11, on three-objective WFG2, the solutions of NSGA-III get close to the PFs than those of NSGA-III-EP, which means the proposed operator slightly decreases the convergence

TABLE IV
MEDIAN AND IQR RESULTS ON PROBLEMS WITH IRREGULAR PFS OBTAINED BY NSGA-III-EP AND NSGA-III. THE HV INDICATOR IS USED FOR WFG AND DTLZ PROBLEMS AND THE Δ_p INDICATOR IS USED FOR MINUS-DTLZ PROBLEMS. THE BETTER RESULT FOR EACH TEST INSTANCE IS SHOWN WITH DARK GRAY BACKGROUND

Problem	M	NSGA-III-EP	NSGA-III
WFG1	3	$5.22e-1(1.2e-2)$	$5.18e-1(3.9e-2)$ $\star$
	5	$5.41e-1(2.1e-2)$	$6.20e-1(3.6e-2)$ $\circ$
	8	$5.14e-1(5.1e-2)$	$8.00e-1(2.2e-2)$ $\circ$
	10	$6.61e-1(4.5e-2)$	$8.78e-1(9.2e-3)$ $\circ$
	15	$6.40e-1(6.7e-2)$	$8.84e-1(4.1e-3)$ $\circ$
WFG2	3	$9.06e-1(9.9e-3)$	$8.89e-1(6.3e-2)$ $\star$
	5	$9.46e-1(5.4e-3)$	$9.34e-1(4.6e-2)$ $\star$
	8	$9.82e-1(3.7e-3)$	$9.33e-1(7.6e-2)$ $\star$
	10	$9.92e-1(2.9e-3)$	$9.56e-1(6.9e-2)$ $\bullet$
	15	$9.89e-1(3.2e-3)$	$8.53e-1(7.6e-2)$ $\bullet$
WFG3	3	$7.15e-1(1.1e-2)$	$7.41e-1(4.5e-3)$ $\circ$
	5	$7.04e-1(7.8e-3)$	$7.18e-1(5.8e-3)$ $\circ$
	8	$5.92e-1(2.2e-2)$	$5.53e-1(6.5e-2)$ $\bullet$
	10	$5.95e-1(1.6e-2)$	$6.58e-1(2.7e-2)$ $\circ$
	15	$6.02e-1(4.4e-2)$	$6.66e-1(3.9e-2)$ $\circ$
DTLZ5	3	$1.91e-1(1.2e-3)$	$1.87e-1(2.1e-3)$ $\bullet$
	5	$1.10e-1(6.6e-3)$	$9.50e-2(8.5e-3)$ $\bullet$
	8	$9.30e-2(1.9e-3)$	$8.25e-2(9.0e-3)$ $\bullet$
	10	$9.14e-2(6.8e-4)$	$6.60e-2(3.0e-2)$ $\bullet$
	15	$9.11e-2(2.3e-4)$	$8.50e-2(1.5e-2)$ $\bullet$
DTLZ6	3	$1.88e-1(1.8e-3)$	$0.00e+0(0.0e+0)$ $\bullet$
	5	$1.06e-1(6.3e-3)$	$0.00e+0(0.0e+0)$ $\bullet$
	8	$9.10e-2(1.7e-4)$	$0.00e+0(0.0e+0)$ $\bullet$
	10	$8.65e-2(2.0e-2)$	$0.00e+0(0.0e+0)$ $\bullet$
	15	$9.10e-2(3.6e-4)$	$0.00e+0(0.0e+0)$ $\bullet$
DTLZ7	3	$2.47e-1(6.2e-3)$	$2.64e-1(3.3e-3)$ $\circ$
	5	$2.21e-1(7.7e-3)$	$2.48e-1(8.1e-3)$ $\circ$
	8	$1.46e-1(9.5e-3)$	$1.53e-1(7.4e-3)$ $\circ$
	10	$1.37e-1(9.5e-3)$	$1.27e-1(6.6e-3)$ $\bullet$
	15	$1.31e-1(3.9e-3)$	$2.03e-2(6.7e-3)$ $\bullet$
DTLZ1^{-1}	3	$3.60e+1(7.9e-1)$	$2.17e+2(5.7e+1)$ $\bullet$
	5	$8.17e+1(1.3e+0)$	$9.55e+1(1.4e+1)$ $\bullet$
	8	$1.69e+2(1.5e+0)$	$1.72e+2(3.6e+0)$ $\bullet$
	10	$1.72e+2(1.9e+0)$	$1.83e+2(5.0e+0)$ $\bullet$
	15	$1.89e+2(2.1e+0)$	$1.93e+2(1.3e+0)$ $\bullet$
DTLZ2^{-1}	3	$2.77e-1(6.4e-3)$	$2.71e-1(2.0e-2)$ $\star$
	5	$8.19e-1(1.3e-2)$	$8.80e-1(1.2e-1)$ $\star$
	8	$1.93e+0(3.5e-2)$	$2.09e+0(1.0e-1)$ $\bullet$
	10	$2.15e+0(3.9e-2)$	$2.29e+0(6.0e-2)$ $\bullet$
	15	$2.87e+0(2.9e-2)$	$2.61e+0(2.1e-2)$ $\circ$
DTLZ3^{-1}	3	$2.16e+3(9.1e+0)$	$2.18e+3(4.7e+0)$ $\bullet$
	5	$2.07e+3(1.4e+1)$	$2.12e+3(1.2e+1)$ $\bullet$
	8	$2.07e+3(1.5e+1)$	$2.14e+3(1.5e+1)$ $\bullet$
	10	$2.09e+3(1.4e+1)$	$2.16e+3(8.0e+0)$ $\bullet$
	15	$2.13e+3(1.2e+1)$	$2.07e+3(2.4e+1)$ $\circ$
DTLZ4^{-1}	3	$2.71e-1(6.9e-3)$	$2.80e-1(5.7e-2)$ $\star$
	5	$7.81e-1(9.4e-3)$	$7.36e-1(3.9e-2)$ $\circ$
	8	$1.57e+0(4.1e-2)$	$1.90e+0(5.9e-2)$ $\bullet$
	10	$1.73e+0(2.3e-2)$	$2.29e+0(1.1e-1)$ $\bullet$
	15	$2.28e+0(2.3e-2)$	$2.85e+0(5.9e-2)$ $\bullet$
$\bullet$ / $\star$ / $\circ$			29 / 7 / 14

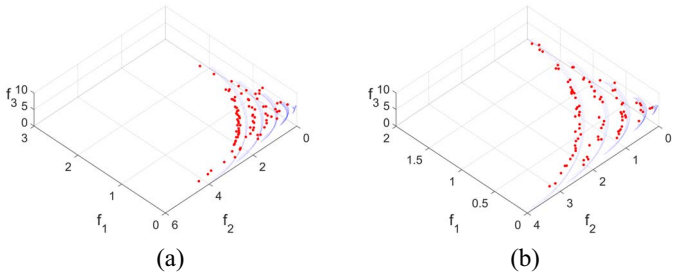


Fig. 11. Final solution sets on the three-objective WFG2. (a) NSGA-III-EP. (b) NSGA-III.

performance. However, on 15-objective WFG2 (shown in Fig. 12), NSGA-III-EP covers larger PF regions than NSGA-III does. This phenomenon is also observed on DTLZ7 as shown in Figs. 13 and 14. For NSGA-III-EP,

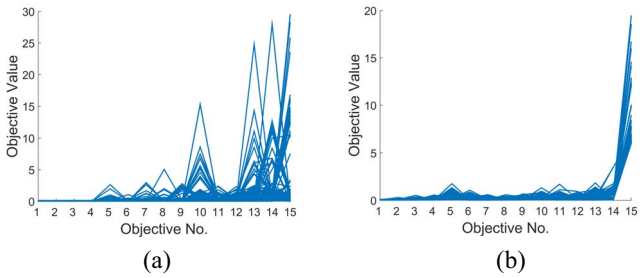


Fig. 12. Final solution sets on the 15-objective WFG2, shown by parallel coordinates. (a) NSGA-III-EP. (b) NSGA-III.

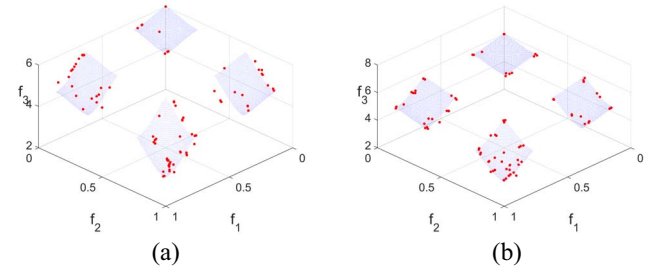


Fig. 13. Final solution sets on the three-objective DTLZ7. (a) NSGA-III-EP. (b) NSGA-III.

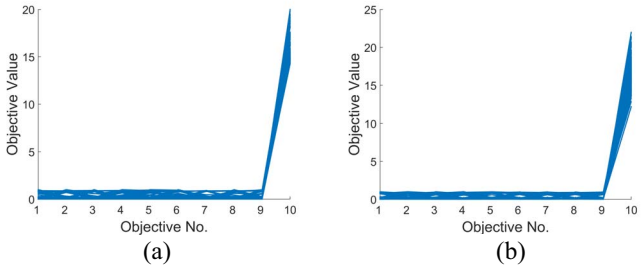


Fig. 14. Final solution sets on the ten-objective DTLZ7, shown by parallel coordinates. (a) NSGA-III-EP. (b) NSGA-III.

some solutions fail to reach the PFs on three-objective test instance but all its solutions are inside the PF regions on ten-objective test instance. Contrarily, it is found that, some solutions of NSGA-III are outside the PF region on the last objective (i.e., [0, 20]) on ten-objective DTLZ7.

3) On problems with complicated PFs such as WFG1, the proposed operator significantly improves the population diversity. On 15-objective WFG1 as shown in Fig. 15, solutions of NSGA-III are only located in small regions while NSGA-III-EP covers much larger regions on the PFs. This is attributed to that NSGA-III-EP finds more boundary solutions than NSGA-III does. This makes the proposed operator have more advantages in high-dimensional objective space since finding the boundary solutions is helpful in preserving diversity.

Besides the PF shape, other factors may also influence the effectiveness of the proposed operator. They are listed as follows.

1) On problems with a huge number of local optima (i.e., DTLZ3 and DTLZ3^{-1}), the proposed operator may

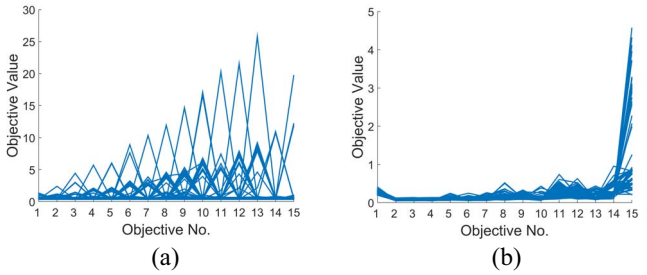


Fig. 15. Final solution sets on the 15-objective WFG1, shown by parallel coordinates. (a) NSGA-III-EP. (b) NSGA-III.

decrease the performance. It is known that a large search step is usually necessary for jumping out of a local optimum. However, when a solution produced by the proposed operator is near a local optimum, its search step is usually decreased.[3] Then, the parameter self-adaptation mechanism may misleadingly pass its corresponding parameters to nearby solutions, thereby causing more solutions being trapped into the local optimum.

2) The effectiveness of proposed operator is related to the algorithm framework. For example, it strengthens MOEA/D on WFG1 but deteriorates the performance of NSGA-III on the same problem. This is not strange since the evolution paths are updated in the environmental selection. Thus, how the environmental selection is performed has direct influence on the effectiveness of the proposed operator.

In fact, the effectiveness of the proposed operator is influenced by the combined effect of different problem properties. For example, it improves the algorithm performance on regular problems WFG5 and WFG9 though they have many local optima. What we can learn from the above experiments is that the proposed operator has good overall performance. When concerning certain problems, traditional operators such as SBX and DE have their own advantages.

B. Comparisons With State-of-the-Art MaOEAs

To further investigate the overall performance of MOEA/D-EP and NSGA-III-EP, we compare them with five state-of-the-art MaOEAs. It should be noted that, this comparison may not be fair since the main contributions of these MaOEAs are their selection operators, and some of them can also be equipped with the proposed operator. As shown in Table VII (in the supplementary material), RVEA performs the best on the WFG test suite in terms of HV value. It obtains the best results on 15 out of 45 test instances. NSGA-III-EP performs the second best and obtains the best results on 11 test instances. MOEA/DD performs very well on three- and five-objective problems, but its performance deteriorates significantly in high-dimensional objective space. VaEA is the only one algorithm which does not use reference vectors, and thus, it shows advantages on degenerate problem WFG3. Neither MOEA/D-EP nor MOEA/D-DU performs well on this test suite. This may be due to their lack of a normalization mechanism. MOMBI2 is slightly better than MOEA/D-EP and MOEA/D-DU, but is still surpassed by other competitors.

Table VIII (in the supplementary material) presents the HV results obtained on the DTLZ test suite. MOEA/D-EP achieves the best overall performance and obtains the best results on 12 test instances. MOEA/DD and MOEA/D-DU also perform well on DTLZ2–DTLZ4. MOMBI2 surpasses all the other algorithms on DTLZ7. Doing well in solving WFG problems, RVEA and VaEA perform relatively poor on this test suite. NSGA-III-EP performs similarly to MOEA/D-EP on DTLZ2 and DTLZ4, but is outperformed on the other problems. Note that, all the five competitors encounter difficulties on DTLZ6. Contrarily, MOEA/D-EP and NSGA-III-EP obtain good results on this problem.

Table IX (in the supplementary material) summaries the Δ_p results on the Minus-DTLZ test suite. Clearly, the five decomposition-based algorithms (MOEA/D-EP, NSGA-III-EP, MOEA/DD, MOEA/D-DU, and RVEA) do not perform well. The reason is that the PF shapes of these problems are inconsistent with the distribution of the reference vectors used in these five algorithms. Hence, searching along the directions of these reference vectors will not generate uniformly distributed solutions. MOMBI2 also employs a set of reference vectors although it is not a decomposition-based algorithm. Consequently, it does not perform well, neither. VaEA gets rid of the above issue because it does not use reference vectors. So it obtains the best performance on this test suite. Apart from VaEA, NSGA-III-EP is the best among the other six algorithms.

The above experiments demonstrate that though MOEA/D and NSGA-III may be considered as "old" algorithms, they are quite competitive with state-of-the-art ones when equipped with the proposed operator. But it is worth pointing out that, no algorithm can beat all the other algorithms on all problems and some algorithms may be more suitable for solving certain problems. In fact, the proposed operator may also be incorporated into these state-of-the-art algorithms. Therefore, there is no conflict between the method in this paper and those used in these competitors.

C. Parameters in the Proposed Operator

Like SBX and DE, the proposed reproduction operator introduces two parameters: 1) S and 2) HL. S is the number of evolution paths for each reference vector while HL is utilized to control F and Cr automatically. In the above experiments, S and HL are fixed to 4 and $\lceil 0.1N \rceil$, respectively. To investigate the performance sensitivity in relation to these two parameters, we test the performance of MOEA/D-EP with different combinations of S and HL values. The detailed sensitivity test is presented in Section S-II in the supplementary material. It is demonstrated that the effectiveness of the operator is less-sensitive to the settings of these parameters. Hence, the above settings (i.e., $S = 4$ and $HL = \lceil 0.1N \rceil$) can serve as a good first choice for the two parameters.

[3]This is indeed a common problem in self-adaptive DE. However, many tricks can be used to alleviate it. For example, one can use a Cauchy distribution in (7) with a certain probability. Since this is not the main contribution of this paper, we do not carry out further investigation. Interested readers may refer to [46] for more details.

V. Conclusion

This paper has presented a new self-adaptive reproduction operator for many-objective optimization, which employs the evolution path to guide the search. Initialization and updating rules of the evolution paths have been proposed to generate potential solutions. Furthermore, a self-adaptation mechanism is introduced to tune the parameters automatically. The implementations in two frameworks have been provided to show that the proposed operator is compatible with well-known MaOEAs. Experimental results demonstrate that the proposed operator is able to significantly enhance the performance of MOEA/D and NSGA-III.

In future work, we will investigate how to perform the proposed operator without employing a set of predefined reference vectors. In addition, implementation into other frameworks (e.g., indicator-based algorithms) may also be considered.

References

[1] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, San Francisco, CA, USA, 2001, pp. 283–290.

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

[3] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evol. Methods Design Optim. Control Appl. Ind. Problems (EUROGEN)*, Athens, Greece, 2001, pp. 95–100.

[4] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[5] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, "Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 169–190, Apr. 2017.

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

[7] S. Chand and M. Wagner, "Evolutionary many-objective optimization: A quick-start guide," *Surveys Oper. Res. Manag. Sci.*, vol. 20, no. 2, pp. 35–42, Dec. 2015.

[8] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 773–791, Oct. 2016.

[9] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao, "Balancing convergence and diversity in decomposition-based many-objective optimizers," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 180–198, Apr. 2016.

[10] Y. Zhou, Z. Chen, and J. Zhang, "Ranking vectors by means of the dominance degree matrix," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 34–51, Feb. 2017.

[11] 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 Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[12] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 440–462, Jun. 2017.

[13] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 450–455, Jun. 2014.

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

[15] Z. Chen, Y. Zhou, and Y. Xiang, "A many-objective evolutionary algorithm based on a projection-assisted intra-family election," *Appl. Soft Comput.*, vol. 61, pp. 394–411, Dec. 2017.

[16] R. B. Agrawal and K. Deb, "Simulated binary crossover for continuous search space," Dept. Mech. Eng., Indian Inst. Technol., Kanpur, India, Rep. IITK/ME/SMD-94027, 1994.

[17] K. Deb and H.-G. Beyer, "Self-adaptive genetic algorithms with simulated binary crossover," *Evol. Comput.*, vol. 9, no. 2, pp. 197–221, Jun. 2001.

[18] R. H. Gómez and C. A. C. Coello, "MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, Jun. 2013, pp. 2488–2495.

[19] R. H. Gómez and C. A. C. Coello, "Improved metaheuristic based on the R2 indicator for many-objective optimization," in *Proc. Annu. Conf. Genet. Evol. Comput. (GECCO)*, Madrid, Spain, 2015, pp. 679–686.

[20] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[21] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Found. Genet. Algorithms*, vol. 2, pp. 187–202, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080948324500180?via%3Dihub

[22] M. Clerc and J. Kennedy, "The particle swarm: Explosion, stability and convergence in multi-dimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 1671–1676, Jan. 2002.

[23] Y. Xiang, Y. Zhou, and H. Liu, "An elitism based multi-objective artificial bee colony algorithm," *Eur. J. Oper. Res.*, vol. 245, no. 1, pp. 168–193, Aug. 2015.

[24] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[25] K. Sindhya, S. Ruuska, T. Haanpää, and K. Miettinen, "A new hybrid mutation operator for multiobjective optimization with differential evolution," *Soft Comput.*, vol. 15, no. 10, pp. 2041–2055, 2011.

[26] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.

[27] M. Daneshyari and G. G. Yen, "Cultural-based multiobjective particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 553–567, Apr. 2011.

[28] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[29] M. Ali, P. Siarry, and M. Pant, "An efficient differential evolution based algorithm for solving multi-objective optimization problems," *Eur. J. Oper. Res.*, vol. 217, no. 2, pp. 404–416, Mar. 2012.

[30] X. Qiu, J.-X. Xu, K. C. Tan, and H. A. Abbass, "Adaptive cross-generation differential evolution operators for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 232–244, Apr. 2016.

[31] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 16–37, Feb. 2016.

[32] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.

[33] A. Zhou, F. Gao, and G. Zhang, "A decomposition based estimation of distribution algorithm for multiobjective traveling salesman problems," *Comput. Math. Appl.*, vol. 66, no. 10, pp. 1857–1868, Dec. 2013.

[34] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, "Generalized decomposition and cross entropy methods for many-objective optimization," *Inf. Sci.*, vol. 282, pp. 363–387, Oct. 2014.

[35] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 838–856, Dec. 2015.

[36] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, Aug. 1998.

[37] P. A. N. Bosman, "On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 51–69, Feb. 2012.

[38] O. Schütze *et al.*, "The directed search method for multi-objective memetic algorithms," *Comput. Optim. Appl.*, vol. 63, no. 2, pp. 305–332, Mar. 2016.

[39] J. A. H. Mejía, O. Schütze, O. Cuate, A. Lara, and K. Deb, "RDS-NSGA-II: A memetic algorithm for reference point based multi-objective optimization," *Eng. Optim.*, vol. 49, no. 5, pp. 828–845, May 2017.

[40] P. A. N. Bosman and E. D. de Jong, "Combining gradient techniques for numerical multi-objective evolutionary optimization," in *Proc. 8th Annu. Conf. Genet. Evol. Comput.*, Seattle, WA, USA, 2006, pp. 627–634.

[41] O. Schütze, V. A. S. Hernández, H. Trautmann, and G. Rudolph, "The hypervolume based directed search method for multi-objective optimization problems," *J. Heuristics*, vol. 22, no. 3, pp. 273–300, Jun. 2016.
[42] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proc. IEEE Int. Conf. Evol. Comput.*, Nagoya, Japan, May 1996, pp. 312–317.
[43] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001.
[44] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evol. Comput.*, vol. 15, no. 1, pp. 1–28, Mar. 2007.
[45] Y.-L. Li *et al.*, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798–1810, Sep. 2015.
[46] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[47] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, Jun. 2013, pp. 71–78.
[48] N. Kowatari, A. Oyama, H. E. Aguirre, and K. Tanaka, "A study on large population MOEA using adaptive ϵ-box dominance and neighborhood recombination for many-objective optimization," in *Proc. 6th Int. Conf. Learn. Intell. Optim. (LION)*, Paris, France, 2012, pp. 86–100, doi: 10.1007/978-3-642-34413-8_7.
[49] M. J. D. Powell, "A survey of numerical methods for unconstrained optimization," *SIAM Rev.*, vol. 12, no. 1, pp. 79–97, Jan. 1970.
[50] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.
[51] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 1. Honolulu, HI, USA, May, 2002, pp. 825–830, doi: 10.1109/CEC.2002.1007032.
[52] Y. Xiang, Y. Zhou, M. Li, and Z. Chen, "A vector angle-based evolutionary algorithm for unconstrained many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 131–152, Feb. 2017.
[53] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[54] O. Schutze, X. Esquivel, A. Lara, and C. A. C. Coello, "Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 504–522, Aug. 2012.
[55] D. A. Van Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations," Ph.D. dissertation, Graduate School Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH, USA, 1999.
[56] Q. Zhang *et al.*, "Multiobjective optimization test instances for the CEC 2009 special session and competition," School Comput. Sci. Elect. Eng., Univ. Essex, Colchester, U.K., Rep. CES-487, Jan. 2008.

Xiaoyu He received the B.Eng. degree from Beijing Electronic Science and Technology Institute, Beijing, China, in 2010, and the M.P.A. degree from the South China University of Technology, Guangzhou, China, in 2016. He is currently pursuing the Ph.D. degree with Sun Yat-sen University, Guangzhou.

His current research interests include evolutionary computation and data mining.

Yuren Zhou received the B.Sc. degree in mathematics from Peking University, Beijing, China, in 1988, and the M.Sc. degree in mathematics and the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 1991 and 2003, respectively.

He is currently a Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His current research interests include design and analysis of algorithms, evolutionary computation, and social networks.

Zefeng Chen received the B.Sc. degree in information and computational science from Sun Yat-sen University, Guangzhou, China, in 2013, and the M.Sc. degree in computer science and technology from the South China University of Technology, Guangzhou, in 2016. He is currently pursuing the Ph.D. degree with Sun Yat-sen University.

His current research interests include evolutionary computation, multiobjective optimization, data mining, and machine learning.