Adaptive Composite Operator Selection and Parameter Control for Multiobjective Evolutionary Algorithm

Qiuzhen Lin[1], Zhiwang Liu[1], Qiao Yan[1], Zhihua Du[1]*, Carlos A. Coello Coello[2], Zhengping Liang[1], Wenjun Wang[1], Jianyong Chen[1]
[1] *College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, P.R. China*
[2] *CINVESTAV-IPN, Department of Computer Science, Mexico, D.F., 07360, Mexico*

Abstract:
The multiobjective evolutionary algorithm based on decomposition (MOEA/D) has shown a superior performance in tackling some complicated multiobjective optimization problems (MOPs). However, the use of different evolutionary operators and their various parameter settings has a significant impact on its performance. To enhance its algorithmic robustness and effectiveness, this paper proposes an adaptive composite operator selection (ACOS) strategy for MOEA/D. Four evolutionary operator pools are used in ACOS and their advantages are combined to provide stronger exploratory capabilities. Regarding each selected operator pool, an online self-adaptation for the parameters tuning is further employed for performance enhancement. When compared with other adaptive and improved strategies designed for MOEA/D, our proposed algorithm is found to be effective and competitive in solving several complicated MOPs.
Keywords: Adaptive composite operator selection; adaptive parameters tuning; differential evolution; decomposition

1. Introduction

Multiobjective optimization problems (MOPs) widely exist in many scientific and engineering applications, which are aimed at optimizing several (often conflicting) objectives simultaneously [10, 13, 40]. No single solution can find the optimum for all the objectives simultaneously due to the fact that the enhancement of one objective may result in the deterioration of another one. Therefore, the target of MOPs is to find a set of equally-optimal solutions, called *Pareto-optimal set* (*PS*), which can be provided to the decision maker as the alternative solutions for various application cases.

Nature-inspired heuristic algorithms, such as evolutionary algorithms (EAs) [11, 48, 59], artificial immune algorithms [5, 33-34, 43] and particle swarm optimization algorithms [8, 51], have shown the promising performance in tackling MOPs. Due to their population-based nature, they are suitable for solving MOPs because, if properly manipulated, they can generate multiple Pareto-optimal solutions in a single run. Particularly, during the last decades, numbers of multiobjective evolutionary algorithms (MOEAs) have been proposed [2, 6, 14, 26, 35, 58]. Most MOEAs are designed based on the use of the Pareto dominance relationship or a decomposition approach [16]. As the Pareto dominance relationship is very simple and straightforward to apply, Pareto-based MOEAs were the most popular in the specialized literature during many years [50]. The most popular MOEAs, *e.g.*, NSGA-II [11] and SPEA2 [60], were all designed based on the Pareto dominance relationship. Until now, there are still

* Corresponding author
Email address: qiuzhlin@szu.edu.cn(Q.Z. Lin), duzh@szu.edu.cn (Z.H. Du)
Phone: +86-75586933530, Fax: +86-75526534078 (Z.H. Du)

many improved Pareto-based MOEAs reported in the literature [12, 21, 27, 31-32, 46]. However, as pointed out in [36, 53], Pareto-based MOEAs have some difficulties to approach the true *Pareto-optimal front* (*PF*) when tackling some complicated MOPs. Thus, a novel MOEA based on decomposition (MOEA/D) was proposed in [36, 53]. It decomposes MOPs into a set of single-objective optimization subproblems (SOPs) and then optimizes all the SOPs cooperatively. The objective of each subproblem is a (linear or nonlinear) weighted aggregation of all the objectives in a MOP. Neighborhood relationships among these subproblems are defined based on the Euclidean distances of their aggregation weight vectors and they can be exploited to enhance the performance of MOEA/D.

Due to the superior performance provided by MOEA/D in solving some complicated MOPs, many enhanced strategies such as dynamical resource allocation [39, 55], enhanced evolutionary operators [36-37, 45], adaptive control methods [29, 47, 57], and matching strategies [28, 30], have been designed based on the framework of MOEA/D. On the dynamical resource allocation, MOEA/D-DRA [55] was proposed based on the fact that different subproblems may have different computational difficulties. This approach designs a dynamic computational resource allocation strategy to assign more computational resources to the non-convergent subproblems. Another dynamic resource allocation scheme for MOEA/D was investigated in [39] to reward the better crossover operator. In this approach, the better one between the simplex crossover operator and the center of mass crossover operator can gain more computational resources. About the enhanced evolutionary operators, differential evolution (DE) was used in [36, 45] to replace simulated binary crossover for effectively producing the new trial vectors, while an opposition-based learning strategy was employed in [37] to accelerate the convergence speed during the evolutionary process. Regarding the adaptive control methods designed in MOEA/D variants, a new version of MOEA/D with an ensemble of different neighborhood sizes (ENS-MOEA/D) was proposed in [57] to decrease the impact of neighborhood size on the performance of MOEA/D. This approach dynamically determines the selection of different neighborhood sizes using their previous search experience, and consequently, this online self-adaptation strategy significantly improves the performance of MOEA/D. In [47], an adaptive DE for MOPs (ADEMO/D) was reported. This approach adopts probability matching and adaptive pursuit as two adaptive strategy selection principles. A DE mutation strategy is picked up from a candidate's DE pool according to a probability that depends on the successful rate to produce better solutions. To adaptively select the preferred recombination operator, a novel bandit-based adaptive operator selection was presented for MOEA/D (MOEA/D-FRRMAB) in [29]. In this approach, the application rates of different DE operators are decided dynamically by their recent performance. A sliding window is used to track the dynamics of the search process by recording the recent fitness improvement rates of different operators, and a decay mechanism is employed to raise the selection probability of the best operator. At last, considering the matching strategies designed for solutions and subproblems, a stable matching model has been proposed for MOEA/D (MOEA/D-STM) in [28]. This approach assigns each promising solution to a subproblem according to the respective preferences. It maintains the good convergence speed and population diversity, and outperforms other enhanced MOEA/D algorithms, such as ENS-MOEA/D and MOEA/D-FRRMAB. Similarly, an improved inter-relationship model [30] was built to match the solutions and subproblems based on their mutual-preferences. Different from the stable matching

model that aims to produce a trade-off between convergence and population diversity, it is essentially a diversity first and convergence second strategy, which enables superior solutions to explore the entire *PF*.

Moreover, some weight generation strategies [17, 23, 41] were also designed to achieve a better approximation for complex Pareto-optimal fronts (PFs). Unlike traditional MOEA/D algorithms that decompose MOPs into a set of subproblems, a new MOEA/D algorithm [4, 7] was proposed to decompose the objective space into different sub-objective spaces using numbers of distinct direction vectors. Each sub-objective space at least owns a solution in order to maintain properly the population diversity. This idea of decomposition using direction vectors was also studied in [22] to combine with a co-evolutionary algorithm, giving rise to the so-called DVCMOA. To extend MOEA/D for high-dimensional MOPs, a generalized decomposition approach was designed in [15], while a systematic sampling approach was presented in [1] to generate uniformly distributed reference points coupled with two independent distance measures and a simple preemptive distance comparison scheme.

It is noted that most of the above MOEA/D variants adopt DE coupled with polynomial mutation as their evolutionary operators. However, several research studies on DE operators have revealed that the use of hybridized DE operators provides an enhanced optimization performance and algorithmic robustness for solving different types of SOPs, because the use of single DE operator may present several limitations in tackling some difficult problems characterized by certain complex features [20, 49]. Since a decomposition approach transforms a MOP into a number of SOPs, it is possible that the competitive approaches designed for solving SOPs are also suitable for MOEA/D. Although an adaptive operator selection for MOEA/D was recently investigated in MOEA/D-FRRMAB to enhance its exploratory capability, four basic DE mutation operators (*i.e.*, "DE/rand/1", "DE/rand/2", "DE/current-to-rand/2" and "DE/current-to-rand/1") were selected in MOEA/D-FRRMAB to compose the operator pool. This combination of DE mutation strategies may not lead to optimal performance, as the composite DE operator pools studied in [49] seem to be more competitive. Working on the research direction suggested by MOEA/D-FRRMAB, this paper proposes an adaptive composite operator selection and parameter control strategy for MOEA/D (called MOEA/D-CDE). The core idea is to design an adaptive MOEA/D algorithm with superior performance. Four composite DE operator pools are adaptively employed (such operators were selected based on their previously reported performance), and their recent fitness improvement rates are stored using a sliding window. An adaptive control strategy is also designed to adjust the parameters in each composite DE pool. Our experimental results validate that MOEA/D-CDE is able to find a good approximated subset of *PS* when solving several complicated MOPs, *e.g.*, the Unconstrained Functions (UF) adopted at the competition held at the *2009 IEEE Congress on Evolutionary Computation* (CEC'2009) [56] and the Walking Fish Group (WFG) problems [18]. When compared with other enhanced variants of MOEA/D, *e.g.*, MOEA/D-DE [36], MOEA/D-DRA [55], ENS-MOEA/D [57], MOEA/D-FRRMAB [29] and MOEA/D-STM [28], MOEA/D-CDE performs best on most of the UF and WFG test problems. The advantages of our proposed adaptive composite operator selection and parameter control strategy are also experimentally analyzed.

The rest of this paper is organized as follows. In Section 2, the related background of our work is presented, such as the mathematical description of MOPs, a brief introduction of a decomposition approach and MOEA/D-DRA. Section 3 gives the details of MOEA/D-CDE, including the adopted composite DE mutation strategies, the adaptive composite operator selection, and the adaptive parameter control strategy. The experimental results of MOEA/D-CDE and the corresponding analysis are provided in Section 4. At last, the conclusions and future work are summarized in Section 5.

2. Related Background

2.1 Multiobjective Optimization Problems

Unconstrained multiobjective optimization problems (MOPs) can be stated as follows [40].

$$\min_{\vec{x}\in\Omega} F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_m(\vec{x}))^T \quad (1)$$

where Ω is the decision (variable) space, $\vec{x} = (x_1, x_2, ..., x_n) \in \Omega$ is a candidate solution with n variables, $F : \Omega \to R^m$ defines m real-valued objective functions and R^m is called the objective space. In many real-world applications, no point in Ω can minimize all the objectives simultaneously. The best trade-offs among the objectives can be attained by using the definition of Pareto optimality [10, 13, 14].

Definition 1 (Pareto-dominance): A decision variable vector $\vec{x}$ is said to dominate another decision variable vector $\vec{y}$ (noted as $\vec{x} \prec \vec{y}$) if and only if

$$(\forall i \in \{1, 2, ..., m\} : f_i(\vec{x}) \le f_i(\vec{y})) \wedge (\exists j \in \{1, 2, ..., m\} : f_j(\vec{x}) < f_j(\vec{y})) \quad (2)$$

Definition 2 (Pareto-optimal): A solution $\vec{x}$ is said to be Pareto-optimal if and only if

$$\neg\exists \vec{y} \in \Omega : \vec{y} \prec \vec{x} \quad (3)$$

Definition 3 (Pareto-optimal set): The set PS includes all the Pareto-optimal solutions, as defined by

$$PS = \{\vec{x} \mid \neg\exists \vec{y} \in \Omega : \vec{y} \prec \vec{x}\} \quad (4)$$

Definition 4 (Pareto-optimal front): The set PF includes the values of all the objective functions corresponding to the Pareto-optimal solutions in PS.

$$PF = \{F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_m(\vec{x}))^T \mid \vec{x} \in PS\} \quad (5)$$

2.2 Decomposition Approach

There are several approaches that can be used to decompose MOPs into a number of SOPs, such as the weighted sum approach, the Tchebycheff approach and the boundary intersection method [40, 53-54]. In this paper, the Tchebycheff approach is adopted as it is mostly used in many variants of MOEA/D [28-30, 36, 45, 53, 55, 57]. This approach is formulated as follows.

$$\begin{aligned} &\text{minimize } g^{tch}(\vec{x} \mid \vec{w}, \vec{z}^*) = \max_{1\le i\le m} \{w_i \mid f_i(\vec{x}) - z_i^* \mid\} \\ &\text{subject to } \vec{x} \in \Omega \end{aligned} \quad (6)$$

where Ω is the decision (variable) space, $\vec{z}^* = (z_1^*, z_2^*, ..., z_m^*)$ is a vector of reference point, *i.e.*, $z_i^* = \min\{f_i(\vec{x}) \mid \vec{x} \in \Omega\}$ for each $i = 1, 2, ..., m$. It is noted that when w_i is set to 0, it will be replaced by $w_i = 10^{-6}$ in Eq. (6). For each Pareto optimal point $\vec{x}$, there exists a weight vector $\vec{w}$ to make sure that $\vec{x}$ is also the optimal solution of Eq. (6). Actually, each optimal solution of Eq. (6) is also a Pareto optimal solution of Eq. (1). Therefore, all the Pareto optimal solutions can be obtained by using

a set of uniformly distinct weight vectors.

2.3 Basic Differential Evolution Operators

Differential evolution is very suitable for dealing with continuous optimization problems [3, 20, 25, 32, 38, 44]. It generally works through a simple cycle by using mutation, crossover and selection operators, which are respectively introduced as follows.

(1) Mutation: differential evolution employs a mutation strategy to generate a mutant vector $\vec{v}_{i,g}$ with respect to each individual $\vec{x}_{i,g}$ (called a target vector) at generation g. The most widely used DE mutation strategies include "DE/rand/1", "DE/rand/2", "DE/best/1", "DE/current-to-rand/1", "DE/current-to-best /2" and "DE/rand-to-best/1" [42]. For example, the basic strategy "DE/rand/1" can be defined as follows.

$$\vec{v}_{i,g} = \vec{x}_{r_1,g} + F \times (\vec{x}_{r_2,g} - \vec{x}_{r_3,g}) \tag{7}$$

where F is called the mutation scaling factor, $\vec{x}_{r_1,g}$, $\vec{x}_{r_2,g}$ and $\vec{x}_{r_3,g}$ are three distinct individuals that are randomly selected from the evolved population at the current generation.

(2) Crossover: after the mutant vector is produced, the crossover operator is further applied. The most widely used recombination operator in DE is binomial crossover, which can be outlined as follows.

$$u_{i,j,g} = \begin{cases} v_{i,j,g}, & \text{if}(r \le CR \text{ or } j = j_{rand}) \\ x_{i,j,g}, & \text{otherwise} \end{cases} \tag{8}$$

where $CR \in [0,1]$ is called the crossover rate, r is a uniformly distributed random number in $[0,1]$, j_{rand} is a random index selected from $\{1,2,...,n\}$ (n is the number of decision variables) to make sure that at least one variable $u_{i,j,g}$ ($j \in [1,n]$) of $\vec{u}_{i,g}$ is inherited from $\vec{v}_{i,g}$. It is noted that $u_{i,j,g}$ will be reinitialized within the feasible range when it is outside its allowable bounds. For example, $u_{i,j,g}$ is reset to $r \times (b_j - a_j)$ when $u_{i,j,g} > b_j$ or $u_{i,j,g} < a_j$, where r is a uniformly distributed random number in $[0,1]$, a_j and b_j are respectively the lower and upper bounds of the j-th decision variable.

(3) Selection: the selection operation is conducted by comparing the target vector $\vec{x}_{i,g}$ with the trial vector $\vec{u}_{i,g}$. The better one is usually selected to survive in the next generation. Generally, it can be defined as follows.

$$\vec{x}_{i,g+1} = \begin{cases} \vec{u}_{i,g}, & \text{if}(f(\vec{u}_{i,g}) < f(\vec{x}_{i,g})) \\ \vec{x}_{i,g}, & \text{otherwise} \end{cases} \tag{9}$$

It has been experimentally found that "DE/rand/1" and "DE/rand/2" have a slow convergence speed but strong exploratory capabilities to avoid premature convergence. Thus, they are suitable for solving multimodal problems. On the other hand, "DE/best/1", "DE/best/2", "DE/rand-to-best/1" and "DE/current-to-best/1" present a fast convergence speed as they employ the best solution found so far to do further exploration. Consequently, they are good at tackling unimodal problems [42]. "DE/current-to-rand/1" is a rotation-invariant strategy, which makes it more suitable to solve rotated problems when compared to other DE strategies [20]. That is to say, different DE mutation strategies have certain particular features, which may behave differently in solving different types of SOPs [49]. Therefore, in this paper, we use multiple DE mutation strategies to integrate the composite operator

pools, with the aim of combining their advantages by using an adaptive operator selection strategy. In this way, our proposed algorithm can solve, in a better way, different types of subproblems that are decomposed from different types of MOPs.

2.4 Review of the baseline algorithm MOEA/D-DRA

In this paper, MOEA/D-DRA [55] which won the CEC2009 multiobjective algorithm contest is the baseline algorithm on which MOEA/D-CDE is based. By embedding the proposed adaptive composite operator selection and parameter control strategy into MOEA/D-DRA, we introduce here MOEA/D-CDE. Let's assume that N uniformly distributed weight vectors $\vec{w}=\{\vec{w}_1,\vec{w}_2,...,\vec{w}_N\}$ are available, where each $\vec{w}_j=\{w_{j,1},w_{j,2},...,w_{j,m}\}$ ($j=1,2,...,N$) satisfies $\sum_{k=1}^{m} w_{j,k}=1$ and $w_{j,k}\geq 0$ ($k\in[1,m]$). Then, the approximation of PF in Eq. (1) can be decomposed into N scalar optimization subproblems using the Tchebycheff approach (defined in Eq. (6)) with N weight vectors $\vec{w}$. For each weight vector, its *T*-neighborhoods are composed by the set of T closest weight vectors based on their Euclidean distances. During the evolutionary search, MOEA/D-DRA maintains a population of N individuals $P=\{\vec{x}_{1,g},\vec{x}_{2,g},...,\vec{x}_{N,g}\}$, where $\vec{x}_{i,g}$ ($i=1,2,...,N$) respectively represent the potential solutions for the *i*-th subproblem at generation *g*. The best value (*i.e.*, the lowest value for minimization problems) found for each objective in Eq. (1) is stored using z_i^* ($i=1,2,...,m$), *i.e.*, $z_i^*=\min\{f_i(\vec{x}_{1,g}),...,f_i(\vec{x}_{N,g})\}$. As different subproblems may have different computation difficulties, a dynamic computational resource assignment was designed in MOEA/D-DRA to automatically allocate the computational effort for different subproblems. At first, the utility π_i for subproblem i ($i=1,2,...,N$) is computed as follows.

$$\pi_i=\begin{cases}1 & \text{if } \Delta_i>0.001\\ (0.95+0.05\times\Delta_i/0.001)\times\pi_i & \text{otherwise}\end{cases} \tag{10}$$

where Δ_i is the relative decrease of the aggregated value in subproblem i, which is defined by

$$\Delta_i=\frac{g^{tch}(\vec{x}_{old}\mid\vec{w}_i,\vec{z}^*)-g^{tch}(\vec{x}_{new}\mid\vec{w}_i,\vec{z}^*)}{g^{tch}(\vec{x}_{old}\mid\vec{w}_i,\vec{z}^*)} \tag{11}$$

As the utility π_i for subproblem i is periodically updated, $\vec{x}_{old}$ is the old best solution found in the last period, while $\vec{x}_{new}$ is the new best solution produced in the current period. If Δ_i is smaller than 0.001, it indicates that the evolutionary search is stagnated in this period. Thus, the value of π_i will be reduced in order to save computational resources.

The pseudo-code of MOEA/D-DRA can be found in **Algorithm 1**, where δ is a pre-defined probability to select the set of parents, n_r is the maximum number of parents that are replaced by the offspring. In the initialization procedure, as shown in lines 1-2 of **Algorithm 1**, the current generation number *g* is set to 0 at the beginning, and the set *A* includes all the indexes of individuals, *i.e.*, $A=\{1,2,...,N\}$ (*N* is the population size). The utility π_i for each subproblem *i* ($i\in[1,N]$) is initially set to 1. *N* weight vectors $\vec{w}=\{\vec{w}_1,\vec{w}_2,...,\vec{w}_N\}$ are uniformly initialized and the population P with *N* solutions is randomly generated. Each weight vector $\vec{w}_i$ ($i\in[1,N]$) finds its *T* neighbors based on their Euclidean distances and the set $B(i)$ includes all the *T* neighbors of weight vector $\vec{w}_i$. The ideal point $\vec{z}^*$ is obtained by $z_j^*=\min\{f_j(\vec{x})\mid\vec{x}\in P\}$ ($j=1,2,...,m$). After that, MOEA/D-DRA enters into the loop of the evolutionary process as illustrated in lines 4-32.

Algorithm 1: The Pseudo-code of MOEA/D-DRA

1 Set $g = 0$, $A = \{1, 2, ..., N\}$, and $\pi_i = 1$ for each $i = 1, 2, ..., N$;
2 Initialize $\vec{w} = \{\vec{w}_1, \vec{w}_2, ..., \vec{w}_N\}$, $P = \{\vec{x}_{1,q}, \vec{x}_{2,q}, ..., \vec{x}_{N,q}\}$, $B(i) = \{i_1, i_2, ..., i_T\}$, $\vec{z}^* = (z_1^*, z_2^*, ..., z_m^*)$;
3 **while** *stopping criterion is not satisfied*
4 Select *m* indexes of the subproblems whose objectives are respectively *m* objectives $f_i(\vec{x})$ in Eq. (1) to form set *I*; Other $\lfloor N/5 \rfloor - m$ subproblems are chosen by using 10-tournament selection based on π_i, which is then added into *I*;
5 **for** *j* =1 to $|I|$
6 *i* =*I*(*j*);
7 **if** *rand*< δ
8 $E = B(i)$;
9 **else**
10 $E = A$;
11 **end**
12 Set $r_1 = i$ and then select two different indexes r_2, r_3 randomly from *E*;
13 Perform DE operator (Eqs. (7)-(8)) to generate a new solution $\vec{u}_{i,q}$ from $\vec{x}_{r_1,g}, \vec{x}_{r_2,g}, \vec{x}_{r_3,g}$;
14 Execute mutation operator (Eq. (12)) on $\vec{u}_{i,q}$ to produce a new solution $\vec{y}_i$;
15 **for** *k* =1 to *m*
16 **if** $z_k^* > f_k(\vec{y}_i)$
17 $z_k^* = f_k(\vec{y}_i)$;
18 **end**
19 **end**
20 $c = 0$;
21 **while** $c < n_r$ && E is not null
22 Randomly pick an index *k* from *E*;
23 **if** $g^{tch}(\vec{y}_i \mid \vec{w}_k, \vec{z}^*) \le g^{tch}(\vec{x}_{k,q} \mid \vec{w}_k, \vec{z}^*)$
24 Replace $\vec{x}_{k,q}$ with $\vec{y}_i$, and set *c*=*c*+1;
25 **end**
26 Delete *k* from *E*;
27 **end**
28 **end**
29 $g = g + 1$;
30 **if** $\mathrm{mod}(g, 50) == 0$
31 update the utility of each subproblem using Eq. (10);
32 **end**
33 **end**
34 **return** *P*;

During the evolutionary process, the dynamic resource assignment is firstly conducted to select the subproblems for evolution as introduced in line 4, where the chosen subproblems are preserved in the set *I*. Then, for each subproblem *i* in *I*, a uniformly distributed random number *rand* is used to determine the parent set *E* that is selected from the entire population or the *T* neighbors of subproblem *i* (shown in lines 7-11). Each individual in *I* is evolved using the DE and mutation operators as respectively stated in lines 13-14. Assuming that the mutant solution generated with the DE operator using Eqs. (7)-(8) is $\vec{u}_{i,g}$, polynomial mutation is further performed on $\vec{u}_{i,g}$ to produce a new solution $\vec{y}_i$. Each variable $y_{i,k}$ $(k = 1, 2, ..., n)$ of $\vec{y}_i$ is obtained by

$$y_{i,k} = \begin{cases} u_{i,k,g} + \sigma_k \times (b_k - a_k) & \text{if } r < p_m \\ u_{i,k,g} & \text{otherwise} \end{cases} \tag{12}$$

where a_k and b_k are respectively the lower and upper bounds of the *k*-th variable, r is a uniformly random real number in [0,1], p_m is the mutation probability and σ_k is calculated as follows.

$$\sigma_k = \begin{cases} (2\times r)^{1/(\eta+1)} - 1 & \text{if } r < 0.5 \\ 1 - 2\times(1-r)^{1/(\eta+1)} & \text{otherwise} \end{cases} \tag{13}$$

where r is a uniformly random real number in [0,1] and η is the distribution index that controls the mutation scale. It is noted that each variable $y_{i,k}$ $(k = 1, 2, ..., n)$ will be reset to a random value inside its allowable bounds when it falls outside them.

For each newly generated offspring $\vec{y}_i$, its objectives are evaluated and then used to update the values of the ideal point z_k^* $(k = 1, 2, ..., m)$ as described in lines 15-19. In lines 21-27, if the aggregation function value of $\vec{y}_i$ is better than the parents from *E*, it will substitute at most n_r parents to speed up convergence. At last, the utility π_i of each subproblem i $(i \in [1, N])$ is updated for each 50 generations using Eq. (10), as illustrated in lines 30-32. The above evolutionary phase will be repeated until the stopping criterion is satisfied. It is noted that the stopping condition can be generally set as the maximum number of generations or function evaluations. At the end of algorithm, all the solutions in population *P* are reported as the final approximated *PF*.

3. The Proposed Algorithm

3.1 Composite DE Mutation Strategies

As introduced in Section 2.3, it has been experimentally found that different DE mutation strategies have their own advantages in solving different types of SOPs [49]. For example, "DE/rand/1" and "DE/rand/2" are good at solving multimodal problems; "DE/best/1", "DE/best/2", "DE/rand-to-best/1" and "DE/current-to-best/1" are suitable for handling unimodal problems [42]; "DE/current-to-rand/1" is more suitable for tackling rotated problems [20]. The use of only one trial vector generation strategy may have natural limitations in solving certain types of SOPs. Therefore, multiple DE mutation strategies are used to form the composite DE operator pools in this paper. In this way, their advantages can be combined to enhance the exploratory capabilities and algorithmic robustness when tackling a set of subproblems decomposed from different types of MOPs.

In this paper, an adaptive operator selection is employed to dynamically select a mutation operator pool. The mutation operator pools are composed by the four DE mutation strategies, *i.e.*, "DE/rand/1", "DE/rand/2", "DE/current-to-rand/1" and "DE/current-to-rand/2" [42]. The definition of "DE/rand/1" has been given in Eq. (7), while "DE/rand/2", "DE/current-to-rand/1" and "DE/current-to-rand/2" are respectively defined in Eqs. (14-16).

$$\vec{v}_{i,g} = \vec{x}_{r_1,g} + F\times(\vec{x}_{r_2,g} - \vec{x}_{r_3,g}) + F\times(\vec{x}_{r_4,g} - \vec{x}_{r_5,g}) \tag{14}$$

$$\vec{v}_{i,g} = \vec{x}_{i,g} + F\times(\vec{x}_{i,g} - \vec{x}_{r_1,g}) + F\times(\vec{x}_{r_2,g} - \vec{x}_{r_3,g}) \tag{15}$$

$$v_{i,g} = \vec{x}_{i,g} + K\times(\vec{x}_{i,g} - \vec{x}_{r_1,g}) + F\times(\vec{x}_{r_2,g} - \vec{x}_{r_3,g}) + F\times(\vec{x}_{r_4,g} - \vec{x}_{r_5,g}) \tag{16}$$

where $\vec{x}_{i,g}$ is called the target vector and $\vec{v}_{i,g}$ is the mutated vector, the individuals $\vec{x}_{r_1,g}$, $\vec{x}_{r_2,g}$, $\vec{x}_{r_3,g}$, $\vec{x}_{r_4,g}$ and $\vec{x}_{r_5,g}$ are five distinct solutions randomly selected from the evolved population, which are also different from $\vec{x}_{i,g}$. The scaling factors F and K are positive control parameters for weighting the difference vectors.

Due to the fact that the composite DE operator pools have been well studied and are suitable for

solving different types of SOPs [49], this paper extends the idea of composite DE operator pools to tackle MOPs. During the evolution, an operator pool applied for each subproblem will be chosen from the four operator pools, by using the proposed adaptive operator selection strategy introduced in Section 3.2. These four operator pools op_i $(i = 1, 2, 3, 4)$ are given by

(1) op_1 : "DE/rand/1" and "DE/rand/2";

(2) op_2 : "DE/current-to-rand/1" and "DE/rand/2";

(3) op_3 : "DE/current-to-rand/1";

(4) op_4 : "DE/current-to-rand/2".

Algorithm 2: Composite_Operator(op , F , E , $\vec{x}_{i,g}$ **)**

switch (op)
  case op_1 **:**
    Randomly select three distinct indexes r_1, r_2, r_3 from E ;
    Use Eqs. (7) and (8) with F to produce a candidate solution $\vec{y}_1$;
    Randomly select five distinct indexes r_1, r_2, r_3, r_4, r_5 from E ;
    If the aggregation value (Eq.(6)) of $\vec{x}_{r_2,g}$ is larger than $\vec{x}_{r_3,g}$, exchange the values of r_2 and r_3 ;
    If the aggregation value (Eq.(6)) of $\vec{x}_{r_4,g}$ is larger than $\vec{x}_{r_5,g}$, exchange the values of r_4 and r_5 ;
    Use Eqs. (14) and (8) to produce a candidate solution $\vec{y}_2$;
    return $\vec{y}_1, \vec{y}_2$;
  case op_2 **:**
    Randomly select three indexes r_1, r_2, r_3 ($r_1 \neq r_2 \neq r_3 \neq i$) from E ;
    If the aggregation value (Eq.(6)) of $\vec{x}_{r_2,g}$ is larger than $\vec{x}_{r_3,g}$, exchange the values of r_2 and r_3 ;
    Use Eqs. (15) and (8) to produce a candidate solution $\vec{y}_1$;
    Randomly select five distinct indexes r_1, r_2, r_3, r_4, r_5 from E ;
    Use Eqs. (14) and (8) with F to produce a candidate solution $\vec{y}_2$;
    return $\vec{y}_1, \vec{y}_2$;
  case op_3 **:**
    Randomly select three indexes r_1, r_2, r_3 ($r_1 \neq r_2 \neq r_3 \neq i$) from E ;
    Use Eqs. (15) and (8) with F to produce a candidate solution $\vec{y}_1$;
    return $\vec{y}_1$;
  case op_4 **:**
    Randomly select five distinct indexes r_1, r_2, r_3, r_4, r_5 from E ;
    Use Eqs. (16) and (8) with F to produce a candidate solution $\vec{y}_1$;
    return $\vec{y}_1$;
end

The "DE/rand/1" strategy is the most commonly used strategy in many DE variants [3, 19-20, 25, 36, 38, 42, 44-45], where all difference vectors are randomly selected from the evolved population. Consequently, it has no bias to any special search direction, and thus chooses a new search direction at random each time. In the "DE/rand/2" and "DE/current-to-rand/2" strategies, two difference vectors are employed, which may lead to larger perturbation than "DE/rand/1" and "DE/current-to-rand/1" that are coupled with only one difference vector. Thus, they can produce the trial vectors with more significant difference than "DE/rand/1" and "DE/current-to-rand/1". It is noted that in operator pool op_1, the difference vectors $\vec{x}_{r_2,g}$ and $\vec{x}_{r_4,g}$ in "DE/rand/2" are always better than $\vec{x}_{r_3,g}$ and $\vec{x}_{r_5,g}$ by switching their positions. In this way, the base vector will be mutated following the better evolutionary direction to speed up convergence. In the same way, "DE/current-to-rand/1" in operator pool op_2 also employs the above difference vector selection strategy to speed up convergence. To clearly illustrate the implementation of the composite operator pools, the corresponding pseudo-code is given in

Algorithm 2, the input of which is *op* (the selected operator pool), *F* (the adaptive scaling factor introduced in Section 3.3), *E* (the index set of parents), and $\vec{x}_{i,g}$ (the target vector at the current generation). The output of **Algorithm 2** are two offspring ($\vec{y}_1$, $\vec{y}_2$) for op_1 and op_2, and one offspring ($\vec{y}_1$) for op_3 and op_4.

3.2 Adaptive Composite Operator Selection

It has been experimentally found that different DE mutation strategies coupled with different control parameter settings have shown several advantages in solving different types of SOPs [9, 20, 25, 29, 38, 42]. Thus, they can be effectively combined to enhance overall performance. In this paper, four operator pools introduced in Section 3.1 are used to produce new solutions. In order to select a better operator pool, a bandit-based operator pools selection scheme is employed. In this approach, it can adaptively determine the preferred operator pool according to the historical search experience [29]. At first, a two-dimensional performance array $Array(2)(W)$ is used to record the impact caused by the application of an operator pool during the evolutionary process. $Array(1)(j)$ ($j = 1, 2, ..., W$) will mark the indexes ($i = 1, 2, 3, 4$) for the used operator pools op_i, and $Array(2)(j)$ ($j = 1, 2, ..., W$) will memorize the fitness improvement rates (*FIR*) that are obtained with the used operator pools op_i ($i = 1, 2, 3, 4$), as defined in Eq. (17).

$$FIR_i = \frac{pf_i - cf_i}{pf_i} \tag{17}$$

where pf_i is the fitness value of the parent, and cf_i is the fitness value of the offspring. This array follows the first-in, first-out (*FIFO*) mechanism. That is to say, the recently used operator pool and its *FIR* value are added at the tail of this array, while the first one in the array is removed. Based on this array, the $Reward_i$ ($i = 1, 2, 3, 4$) is calculated by summarizing all *FIR* values obtained by each operator pool op_i, and then the credit value FRR_i assigned to the operator pool op_i is obtained as follows.

$$FRR_i = \frac{Reward_i}{\sum_{j=1}^{4} Reward_j} \tag{18}$$

Then, the FRR_i values and the times of operator pool (stored in n_i for op_i ($i \in [1, 4]$)) selected in the recent W applications are employed to pick out a better operator pool using Eq. (19).

$$op_i = \arg\max_{i=\{1,2,3,4\}} \left(FRR_i + C \times \sqrt{\frac{2 \times \ln \sum_{j=1}^{4} n_j}{n_i}} \right) \tag{19}$$

where C is a scaling factor to control the trade-off between the fitness improvement rates and the number of used operator pools. Based on our experiments, too small values or too large values of *C* will have a negative impact on the optimization performance when solving certain types of MOPs. Thus, it is suggested to select a value of *C* within the range [4, 7]. In this study, *C* is set to 5.0 as referred from [29]. Therefore, the operator pool op_i ($i \in [1, 4]$) that can give the maximum value of Eq. (19) will be selected to produce the offspring. It is noted that once the offspring are generated using the selected operator pool, this *FIR* array will be updated and used to calculate a new maximum value of

Eq. (19). By this way, the best operator pool can be adaptively adjusted. The pseudo-code for this operator pool selection mechanism is provided in **Algorithm 3** and its input is the performance array $Array(2)(W)$. Based on the performance of the W most recent applications, **Algorithm 3** will adaptively return a selected operator pool.

Algorithm 3: Operator_Selection($Array(2)(W)$ **)**

```
Initialize Reward_i = 0 for all op_i (i = 1,2,3,4);
Set n_i = 0 (i = 1,2,3,4);
for j = 1 to W
    i = Array(1)(j);
    Reward_i = Reward_i + Array(2)(j);
    n_i++;
end
RewardSum = sum_{i=1}^{4} Reward_i;
for i = 1 to 4
    FRR_i = Reward_i / RewardSum;
end
Select an operator pool op_i (i ∈ [1,4]) using Eq. (19);
return op_i;
```

3.3 Adaptive Parameter Control Strategy

The parameter adaptation schemes in DE have been experimentally found to be very competitive in solving different types of SOPs [20, 42]. In this paper, the adaptive parameter strategy in JADE [52] is modified to tackle MOPs. At each generation, the scaling factor F used in the operator pools is independently generated by

$$F = Cauchy(F', 0.1) \tag{20}$$

where $Cauchy(F', 0.1)$ is a random real number sampled from a Cauchy distribution with location parameter F' and scale parameter 0.1. The value of F will be regenerated if $F \le 0$ or $F \ge 1$. This location parameter F' of the Cauchy distribution is initialized to 0.5 and is then updated at the end of each generation, as follows.

$$F' = w_F \times F' + (1 - w_F) \times mean_{POW}(F_{success}) \tag{21}$$

where w_F is a pre-defined weight factor, and the set $F_{success}$ collects all the successful scaling factors that can generate better trial vectors at each generation. The $mean_{POW}(\cdot)$ function stands for the power mean [20], as given by

$$mean_{POW}(F_{success}) = \sum_{x \in F_{success}} (x^k / |F_{success}|)^{1/k} \tag{22}$$

where $|F_{success}|$ denotes the cardinality of the set $F_{success}$, and k is set to 1.5 as it gives the best results on a wide variety of optimization problems [20]. As revealed by our parameter tuning experiments, a small random perturbation added to the weight term w_F can enhance the performance of MOEA/D-CDE. Thus, w_F is randomly generated from [0.8, 1] in this paper.

It is noted that the adaptive parameter control strategy is only employed for the scaling factor F. All the parameter settings of the DE operator pools are given as follows.

(1) In op_1, the parameter F uses the above adaptive parameter control strategy, while CR is set to

1 for "DE/rand/1". The parameters F and CR are respectively set to 0.2 and 0.8, and K is a uniformly distributed random value generated in [0, 1] for "DE/rand/2".

(2) In op_2, the parameter F uses the above adaptive parameter control strategy, while CR is set to 1 for "DE/rand/2". The parameters F and CR are all set to 1 for "DE/current-to-rand/1".

(3) In op_3, the parameter F uses the above adaptive parameter control strategy, while CR is set to 1 and K is set to a uniformly distributed random value in [0, 1] for "DE/current-to-rand/2".

(4) In op_4, the parameter F uses the above adaptive parameter control strategy, while CR is set to 1 for "DE/current-to-rand/1".

3.4 The Complete Algorithm MOEA/D-CDE

In the above subsections, the four composite DE operator pools used in this paper were introduced in Section 3.1, and then an adaptive composite operator selection described in Section 3.2 was employed to pick out a better operator pool for each individual. Besides that, an adaptive parameter approach (introduced in Section 3.3) was exploited to adjust the scaling factor F in each DE operator pool. These proposed approaches are the main contributions of this paper, which greatly enhance the optimization performance and algorithmic robustness. By embedding the proposed approaches into the pseudo-code of MOEA/D-DRA, we gave rise to our proposed MOEA/D-CDE. To clearly introduce the implementation of MOEA/D-CDE, its pseudo-code is given in **Algorithm 4**.

In the initialization phase, similar to MOEA/D-DRA, some relevant parameters (*i.e.*, g, A, π_i, $\vec{w}$, P, $B(i)$ and $\vec{z}^*$) are initialized in lines 1-2 of **Algorithm 4**. Besides that, a two-dimensional performance array $Array(2)(W)$ used to store the fitness improvement rates and the corresponding index point $Array\_index$ are also initially set in line 3.

During the evolutionary process, the subproblems to be solved are selected into set I using a dynamic resource assignment mechanism, as shown in line 5. For each subproblem i in set I, an operator pool op is selected in lines 8-12, where each operator pool is forced to be used once in the first four selections. Then, the parent set E is chosen in lines 13-17 and the scaling factor F used in operator pool op is sampled by Eq. (20) (line 18). With these parameters (*i.e.*, op, E, F, $x_{i,g}$), the offspring are generated using the composite operator pools as introduced in **Algorithm 2** (line 19), and further mutated using polynomial mutation in line 22. The new offspring ($\vec{y}_1, \vec{y}_2$ generated by op_1 or op_2, and $\vec{y}_1$ produced by op_3 or op_3) are used to update the reference point $\vec{z}^*$ in lines 23-27, and renew at most n_r individuals from the parents set E as shown in lines 28-37, where indexof(op) returns the index of operator pool op. The fitness improvement rates caused by op will be stored in $Array(2)(W)$ (lines 39-40). Once all the individuals in set I are evolved, the successful values of F are used to update the F' value using Eqs. (21)-(22), as shown in line 43. At last, the utility π_i of each subproblem i ($i \in [1, N]$) is updated at each 50 generations using Eq. (10), as illustrated in lines 45-47. The above evolutionary phase will be repeated until reaching the stopping criterion (*e.g.*, a maximum number of generations or function evaluations). At the end of algorithm, all the solutions in population P are reported as the final approximation of the PF.

Algorithm 4: The Pseudo-code of MOEA/D-CDE

Set $g = 0$, $A = \{1,2,...,N\}$, and $\pi_i = 1$ for each $i = 1,2,...,N$;
Initialize $\vec{w} = \{\vec{w}_1, \vec{w}_2, ..., \vec{w}_N\}$, $P = \{\vec{x}_{1,q}, \vec{x}_{2,q}, ..., \vec{x}_{N,q}\}$, $B(i) = \{i_1, i_2, ..., i_T\}$ and $\vec{z}^* = (z_1^*, z_2^*, ..., z_m^*)$;
Set $Array(i)(j) = 0$ for each $i = 1,2$ and $j = 1,2,...,W$, $Array\_index = 1$;
while *stopping criterion is not satisfied*
    Select *m* indexes of the subproblems whose objectives are respectively *m* objectives $f_i(\vec{x})$ in Eq. (1) to form the set *I*; Other $\lfloor N/5 \rfloor - m$ subproblems are chosen by using 10-tournament selection based on π_i, which is then added into *I*;
    for $j = 1$ to $|I|$
        $i = I(j)$;
        if $j \le 4$ && $g == 0$
            $op = op_j$;
        else
            op = **Operator_Selection(** $Array(2)(W)$ **) (Algorithm 3)**;
        end
        if $rand < \delta$
            $E = B(i)$;
        else
            $E = A$;
        end
        The scaling factor *F* used in operator pool op is generated by Eq. (20);
        {$\vec{y}_1'$, $\vec{y}_2'$}=**Composite_Operator(** op, F, E, $\vec{x}_{i,q}$ **)(Algorithm 2)**;// $\vec{y}_2'$ is null when *op* is op_3 or op_4
        for $l = 1$ to 2
            if ($\vec{y}_l'$ is not null)
                Apply mutation operator (Eq. (12)) on $\vec{y}_l'$ to produce a new solution $\vec{y}_l$;
                for $k = 1$ to m
                    if $z_k^* > f_k(\vec{y}_l)$
                        $z_k^* = f_k(\vec{y}_l)$
                    end
                end
                $c = 0$, $E_l = E$;
                while $c < n_r$ && E_l is not null
                    Randomly pick an index *k* from E_l;
                    $\Delta_i = \left(g^{tch}(\vec{x}_{k,q} \mid \vec{w}_k, \vec{z}^*) - g^{tch}(\vec{y}_l \mid \vec{w}_k, \vec{z}^*) \right) \Big/ g^{tch}(\vec{x}_{k,q} \mid \vec{w}_k, \vec{z}^*)$
                    if $\Delta_i \ge 0$
                        Replace $\vec{x}_{k,q}$ with $\vec{y}_l$, and set *c*=*c*+1;
                        $FIR_{\text{indexof}(op)} = FIR_{\text{indexof}(op)} + \Delta_i$;
                    end
                    Delete *k* from E_l;
                end
            end
            $Array[0][Array\_index] = \text{indexof}(op)$ and $Array[1][Array\_index++] = FIR_{\text{indexof}(op)}$;
            $Array\_index = Array\_index \bmod W$;
        end
    end
    Update F' using Eqs. (21)-(22);
    $g = g + 1$;
    if $\text{mod}(g, 50) == 0$
        update the utility of each subproblem using Eq. (10);
    end
end
return *P*;

4 Experimental Studies

In this section, several experimental studies are conducted in order to analyze the advantages of

MOEA/D-CDE. First, some background of our experiments is presented. 19 test instances are then introduced in Section 4.1, and two performance measures are described in Section 4.2 for examining both convergence and population diversity. Section 4.3 provides the parameters settings for all the compared algorithms. Second, MOEA/D-CDE is compared to five competitive MOEA/D variants, *i.e.*, MOEA/D-DE [36], MOEA/D-DRA [55], ENS-MOEA/D [57], MOEA/D-FRRMAB [29] and MOEA/D-STM [28], and the relevant discussions on their performance comparison are given in Section 4.4. Third, the effectiveness of the composite DE operator pools and the proposed adaptive parameter control strategy are respectively analyzed in Sections 4.4 and 4.5. Finally, the time complexity analysis of our algorithm is provided in Section 4.6.

4.1 Test Instances

In order to evaluate the performance of MOEA/D-CDE, 19 unconstrained test instances are employed here as our benchmark problems for conducting the empirical study. They can be classified into two test problem series, *i.e.*, the UF and the WFG problem series. More specifically, UF1-UF10 are the benchmark functions adopted in the CEC2009 competition [56], and they are characterized with very complicated *PS* in decision space; WFG1-WFG9 are designed to have a wide range of complex characteristics [18], including non-separable, deceptive, degenerate problems, mixed *PF* shapes and variable dependencies. UF and WFG test problems are difficult to be optimized, and have already been used in many MOEAs [28-30, 36, 45] to investigate their optimization performance. The number of decision variables for UF1-UF10 is set to 30, while the numbers of position-related and distance-related decision variables for WFG1-WFG9 are respectively set to 8 and 2. Moreover, UF1-UF7 and WFG1-WFG9 are bi-objective test problems, while UF8-UF10 are three-objective test problems. More detailed information about the UF and the WFG test problems can be respectively found in [56, 18].

4.2 Performance Measures and Experimental Settings

4.2.1 Performance Metrics

After the approximation sets are obtained by the compared algorithms, two performance measures are used to assess their performance [59], *i.e.*, inverted generational distance (IGD) and Hypervolume (HV). These performance measures can assess both convergence and diversity for the approximation sets and they are defined next.

IGD: Let P^* be a set of Pareto-optimal points uniformly sampled along the true *PF*, and P be an approximation set obtained by MOEA. The IGD value of P is computed as follows [31].

$$IGD\left(P,P^*\right)=\frac{\sum_{\vec{x}\in P^*} dist\left(\vec{x},P\right)}{|P^*|} \tag{23}$$

where $dist\left(\vec{x},P\right)$ is the Euclidean distance between the point $\vec{x}$ and its nearest neighbor in P, and $|P^*|$ is the cardinality of P^*. The true *PF* of each test problem has to be known in advance when calculating IGD. In our empirical studies, 1000 and 10000 uniformly distributed points are respectively sampled along the *PF* for the bi-objective and the three-objective test instances. Generally, a lower IGD value indicates the better convergence and diversity of P.

HV: Let $\vec{z}^* = (z_1^*, z_2^*, ..., z_m^*)^T$ be a reference point in the objective space that is dominated by all the Pareto-optimal points. HV measures the size of the objective space dominated by the solutions in P and bounded by $\vec{z}^*$, and is defined as follows [58].

$$HV(P) = VOL\left(\bigcup_{\vec{x} \in P} \left[f_1(\vec{x}), z_1^* \right] \times ... \times \left[f_m(\vec{x}), z_m^* \right] \right) \tag{24}$$

where the function *VOL*(·) means the Lebesgue measure. In our experiments, the normalized objective function values are used to compute HV and the reference points are set to (2.0, 2.0) and (2.0, 2.0, 2.0) respectively for bi-objective and three-objective test problems. A larger HV value is always preferred as it indicates a better quality of P for approximating the entire *PF*.

4.2.2 Experimental Settings for the Compared Algorithms

In this study, our algorithm MOEA/D-CDE is compared with respect to five competitive MOEA/D variants, *i.e.*, MOEA/D-DE [36], MOEA/D-DRA [55], ENS-MOEA/D [57], MOEA/D-FRRMAB [29] and MOEA/D-STM [28] on all the UF and WFG test problems. It is noted that the Tchebycheff approach defined in Eq. (6) is used as the decomposition method for all the compared algorithms. The parameters settings of MOEA/D-DE [36], MOEA/D-DRA [55], ENS-MOEA/D [57], MOEA/D-FRRMAB [29] and MOEA/D-STM [28] are respectively listed in Table 1. All these parameters settings used in this paper are suggested by their authors.

Table 1
The parameters settings of the compared algorithms

MOEA/D-DE	$F = 0.5$, $CR = 1.0$, $n_r = 2$.
MOEA/D-DRA	$F = 0.5$, $CR = 1.0$, $T = 0.1 \times N$, $n_r = 0.01 \times N$.
ENS-MOEA/D	$F = 0.5$, $CR = 1.0$, $LP = 50$.
MOEA/D-FRRMAB	$F = 0.5$, $CR = 1.0$, $K = 0.5$, $C = 5.0$, $W = 0.5 \times N$, $D = 1.0$, $n_r = 2$.
MOEA/D-STM	$F = 0.5$, $CR = 1.0$, $n_r = 2$.
MOEA/D-CDE	$n_r = 2$, $W = 0.5 \times N$, $C = 5.0$.

In Table 1, *F* and *CR* are respectively the scaling factor and crossover rate used in DE. T denotes the neighborhood size and n_r indicates the maximum number of parents replaced by each new offspring. For ENS-MOEA/D, *LP* defines the number of generations to update the selection probability for each neighborhood size. The possible neighborhood sizes for bi-objective and three-objective problems in ENS-MOEA/D are respectively set to {30, 60, 90, 120} and {60, 80, 100, 120, 140}. *K* is also a scaling factor used in MOEA/D-FRRMAB. *C*, *W* and *D* are the three control parameters used in the bandit-based adaptive operator selection of MOEA/D-FRRMAB. For all the compared algorithms, the population size N is set to 600 for UF1-UF7 and 1000 for UF8-UF10, while N is set to 100 for WFG1-WFG9; the maximum number of function evaluations is set to 600000 for the UF test problems and to 25000 for the WFG test problems. All the compared algorithms are implemented in JAVA, except for ENS-MOEA/D which was implemented in MATLAB. The parameters settings of our proposed MOEA/D-CDE algorithm are further clarified as follows. The probability for polynomial mutation is set as $p_m = 1/n$, and its distribution index is set to 20, *i.e.*, $\eta = 20$. The settings of the control parameters *CR*, *F* and *K* in the four operator pools adopted are explained in Section 3.3. The neighborhood size T is set to 20 and n_r is set to 2. The probability δ that controls the selection of

parent vectors from the entire population or the neighborhood is set to 0.9. As the size of the performance array W used in the adaptive composite operator selection strategy will lower the optimization performance when it is set to a very small value, it is suggested to set W not smaller than $0.5\times N$ (we set W as $0.5\times N$ in our experiments). Each compared algorithm is run by 30 independent times for each test problem. Experimental results are collected in the corresponding comparison tables, and the best mean results of IGD and HV are highlighted in boldface. Moreover, in order to have a statistically sound conclusion, a popular nonparametric test (*i.e.*, Wilcoxon's rank sum test) was further conducted to assess the statistical significance of the difference between the results obtained by MOEA/D-CDE and those obtained by the other algorithms with a significance level α=0.05.

4.3 Comparison of MOEA/D-CDE with Five MOEA/D Variants

4.3.1 Performance Comparison on UF instances

In this subsection, MOEA/D-CDE is compared with respect to five MOEA/D variants, namely, MOEA/D-DE, MOEA/D-DRA, ENS-MOEA/D, MOEA/D-FRRMAB and MOEA/D-STM, on all the UF test problems.

Table 2
Comparative results of all the algorithms adopted on the UF test problems regarding IGD

Problems \ Algorithms		MOEA/D-DE	MOEA/D-DRA	ENS-MOEA/D	MOEA/D-FRRMAB	MOEA/D-STM	MOEA/D-CDE
UF1	Mean	**7.759E-04** +	1.006E-03 -	8.734E-04 -	8.094E-04 ≈	8.721E-04 -	8.072E-04
	Std	**5.33E-05**	5.51E-05	5.32E-05	5.88E-05	6.23E-05	2.83E-05
UF2	Mean	1.946E-03 -	1.698E-03 -	2.241E-03 -	1.171E-03 -	1.473E-03 -	**9.283E-04**
	Std	6.49E-04	1.63E-04	5.13E-04	3.40E-04	8.66E-04	**1.31E-04**
UF3	Mean	3.134E-03 -	**9.659E-04** +	1.437E-03 ≈	1.098E-03 +	2.509E-03 -	1.443E-03
	Std	2.59E-03	**8.30E-05**	2.15E-03	2.95E-04	2.55E-03	8.10E-04
UF4	Mean	5.151E-02 -	5.540E-02 -	4.831E-02 -	4.893E-02 -	4.775E-02 -	**3.190E-02**
	Std	2.92E-03	3.80E-03	2.81E-03	2.91E-03	3.16E-03	**6.77E-04**
UF5	Mean	1.943E-01 -	1.963E-01 -	2.680E-01 -	1.918E-01 -	1.937E-01 -	**9.101E-02**
	Std	5.25E-02	7.05E-02	5.15E-03	5.98E-02	5.97E-02	**2.73E-02**
UF6	Mean	7.186E-02 -	1.676E-01 -	6.133E-02 -	9.073E-02 -	7.030E-02 -	**5.903E-02**
	Std	3.09E-02	1.89E-01	4.31E-02	1.31E-01	3.52E-02	**6.39E-03**
UF7	Mean	8.940E-04 -	1.687E-03 -	8.714E-04 -	8.906E-04 -	8.855E-04 -	**8.006E-04**
	Std	7.80E-05	2.50E-03	4.24E-04	9.71E-05	4.43E-05	**1.66E-04**
UF8	Mean	4.002E-02 -	3.260E-02 -	3.205E-02 -	3.022E-02 ≈	**1.816E-02** +	2.908E-02
	Std	5.45E-03	2.81E-03	2.63E-03	2.29E-03	**4.67E-04**	2.53E-03
UF9	Mean	3.027E-02 -	1.092E-01 -	5.676E-02 -	3.201E-02 -	**1.617E-02** +	2.154E-02
	Std	3.03E-02	4.93E-02	3.15E-02	3.56E-02	**4.66E-04**	5.66E-02
UF10	Mean	3.713E-01 -	3.143E-01 -	3.138E-01 -	3.432E-01 -	3.221E-01 -	**2.168E-01**
	Std	5.23E-02	7.77E-02	4.32E-02	4.93E-02	4.00E-02	**4.31E-02**
-/+/≈		9/1/0	9/1/0	9/0/1	7/1/2	8/2/0	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to the Wilcoxon's rank sum test at a 0.05 significance level

Table 2 provides the comparative results of all the algorithms adopted, in which the mean IGD values and the standard deviation (Std) from 30 independent runs are listed. These simulation results

show that MOEA/D-CDE performs best on UF2, UF4-UF7 and UF10 when compared to the other algorithms. Particularly, MOEA/D-CDE performs better than MOEA/D-DE, MOEA/D-DRA, ENS-MOEA/D, MOEA/D-FRRMAB and MOEA/D-STM on 9, 9, 9, 7 and 8 out of 10 UF test problems, while MOEA/D-CDE is only worse than MOEA/D-DE on UF1, worse than MOEA/D-DRA on UF3, worse than MOEA/D-FRRMAB on UF3, and worse than MOEA/D-STM on UF8 and UF9. Moreover, the Wilcoxon's rank sum test reveals that MOEA/D-CDE performs similarly to ENS-MOEA/D on UF3 and to MOEA/D-FRRMAB on UF1 and UF8. Based on the summary in the last row of Table 2, it is reasonable to conclude that MOEA/D-CDE is better than MOEA/D-DE, MOEA/D-DRA, ENS-MOEA/D, MOEA/D-FRRMAB and MOEA/D-STM when considering all the UF test problems with respect to IGD. This clearly indicates that our proposed approach enhances the optimization performance of MOEA/D-CDE.

Table 3
Comparative results of all the algorithms adopted on the UF test problems regarding HV

Problems \ Algorithms		MOEA/D-DE	MOEA/D-DRA	ENS-MOEA/D	MOEA/D-FRRMAB	MOEA/D-STM	MOEA/D-CDE
UF1	Mean	3.6597 -	3.6613 -	3.6529 -	3.6624 ≈	3.6635 ≈	**3.6645**
	Std	1.59E-03	1.08E-03	1.01E-03	1.57E-03	6.59E-04	**8.54E-04**
UF2	Mean	3.6552 -	3.6558 -	3.6502 -	3.6565 -	3.6580 ≈	**3.6643**
	Std	4.74E-03	8.15E-03	5.18E-03	7.71E-03	2.35E-03	**5.42E-03**
UF3	Mean	3.6401 -	**3.6653** +	3.6598 ≈	3.6234 -	3.6404 -	3.6636
	Std	2.40E-02	**1.53E-02**	1.56E-03	7.63E-02	2.59E-02	4.57E-03
UF4	Mean	3.1930 -	3.1560 -	3.1692 -	3.1717 -	3.2018 -	**3.2438**
	Std	1.20E-02	2.33E-02	1.90E-02	1.58E-02	7.98E-03	**2.14E-02**
UF5	Mean	3.1658 -	2.7735 -	2.6537 -	2.7587 -	**3.2919** +	3.2631
	Std	3.27E-01	3.81E-01	3.04E-01	3.06E-01	**3.55E-01**	3.28E-01
UF6	Mean	3.0686 -	2.9021 -	3.1105 -	2.9839 -	3.1339 -	**3.2465**
	Std	1.04E-01	3.22E-01	2.15E-01	3.25E-01	1.05E-01	**1.38E-01**
UF7	Mean	3.4889 -	3.4807 -	3.4817 -	3.4898 -	3.4945 ≈	**3.4968**
	Std	4.35E-03	4.15E-02	2.08E-01	6.16E-03	1.40E-03	**3.25E-03**
UF8	Mean	7.3543 -	7.3627 -	7.3792 -	7.3828 ≈	**7.4325** +	7.3927
	Std	1.53E-02	2.12E-02	1.03E-02	1.15E-02	**3.12E-02**	2.02E-02
UF9	Mean	7.5523 -	7.3836 -	7.5623 -	7.6482 ≈	**7.6852** ≈	7.6828
	Std	1.52E-01	2.03E-01	1.32E-01	1.63E-01	**2.28E-01**	1.41E-01
UF10	Mean	3.5327 -	3.7621 -	3.8123 ≈	3.6212 -	2.8217 -	**3.8237**
	Std	2.98E-01	2.51E-01	5.82E-01	3.38E-01	5.12E-01	**2.51E-01**
-/+/≈		10/0/0	9/1/0	8/0/2	7/0/3	4/2/4	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to the Wilcoxon's rank sum test at a 0.05 significance level

Table 3 provides the comparative results of all the algorithms adopted on the UF test problems using HV. It is noted that these HV results in Table 3 shows a slight difference with the IGD results in Table 2. This is possible as pointed out in [24] that the performance metrics of IGD and HV show high consistencies on convex PFs and certain contradictions on concave PFs. As observed from Table 3, MOEA/D-CDE performs best on 6 (*i.e.*, UF1-UF2, UF4, UF6-UF7 and UF10) out of 10 UF test problems. In more detail, MOEA/D-CDE performs better than MOEA/D-DE, MOEA/D-DRA, ENS-MOEA/D, MOEA/D-FRRMAB and MOEA/D-STM on 10, 9, 8, 7 and 4 out of 10 UF test problems. Nevertheless, MOEA/D-CDE is beaten by MOEA/D-DRA on UF3 and MOEA/D-STM on

UF5 and UF8. The Wilcoxon's rank sum test indicates that MOEA/D-CDE obtains statistically similar results to ENS-MOEA/D on UF3 and UF10, to MOEA/D- FRRMAB on UF1, UF8 and UF9, and to MOEA/D-STM on UF1-UF2, UF7 and UF9. As summarized in the last row of Table 3, MOEA/D-CDE is better than or similar to MOEA/D-DE, ENS-MOEA/D and MOEA/D-FRRMAB on all the UF test problems. Regarding the comparison with MOEA/D-DRA and MOEA/D-STM, MOEA/D-CDE performs better or similarly on more than half of the UF test problems. Therefore, the advantages of MOEA/D-CDE are further confirmed by using HV.

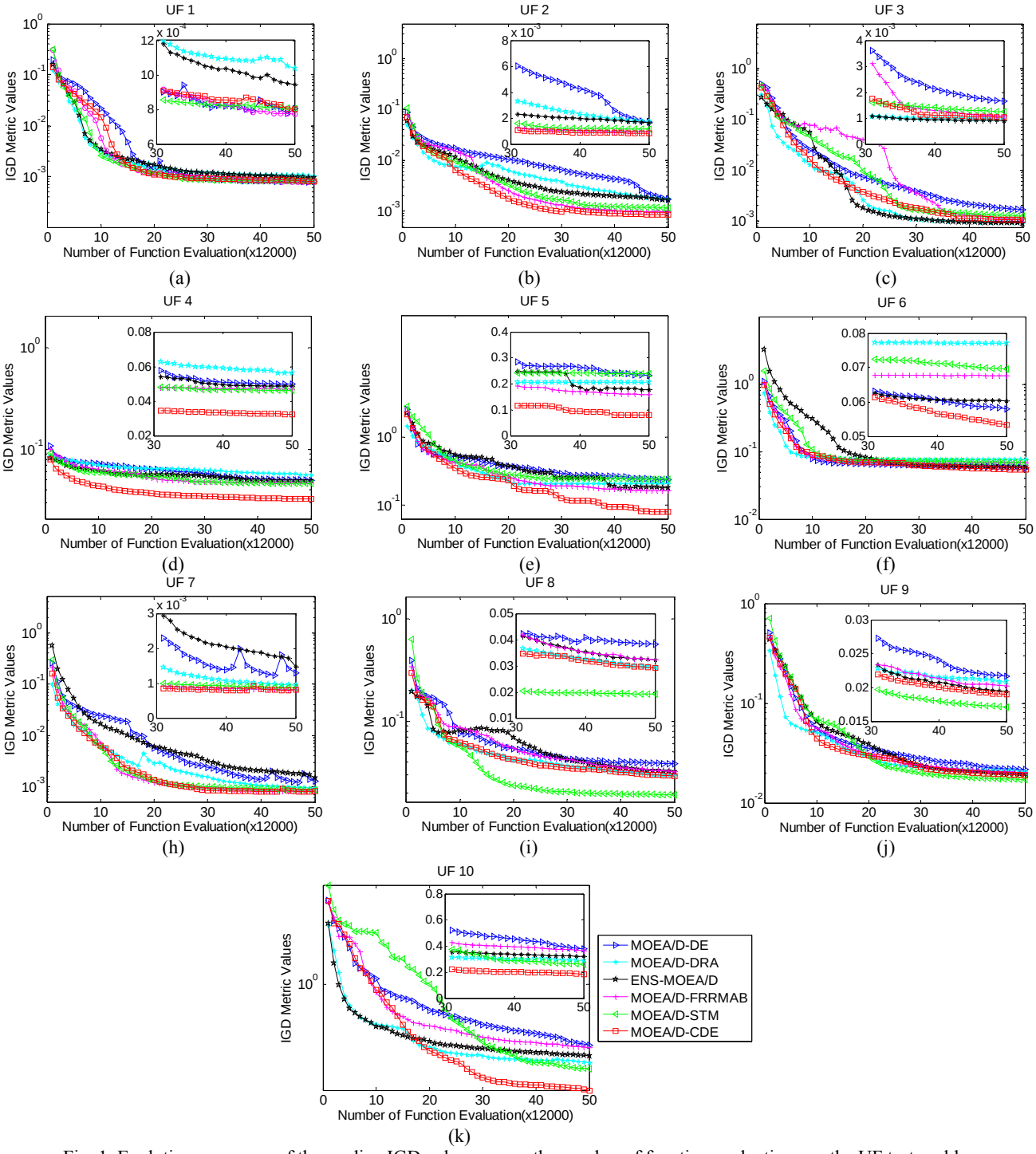

Fig. 1. Evolutionary curves of the median IGD values versus the number of function evaluations on the UF test problems.

In order to provide an overview of the evolutionary progress for all the compared algorithms, Figure 1 plots the evolutionary curves of the median IGD values versus the number of function evaluations on each UF test problem. The plots in Fig. 1 illustrate that MOEA/D-CDE gradually reduces the median IGD values and gets closer to the true *PF* as we approach the maximum number of generations. One important phenomenon observed from Fig. 1 is that MOEA/D-CDE may perform worse at early stages of the evolutionary process, but can outperform the other algorithms at later stages. This could be due to the fact that MOEA/D-CDE uses an adaptive operator selection mechanism to pick up a better set of operators from our four composite operator pools. In this way, the exploratory capability of MOEA/D-CDE is sufficiently enhanced to avoid the intrinsic limitation of using a single DE operator, which may produce stagnation at later stages of the evolutionary process. From Fig. 1, it is clear that MOEA/D-CDE is significantly better than the other algorithms on UF2, UF4, UF5 and UF10. Even for UF1, UF3, UF6, and UF7, MOEA/D-CDE also shows a competitive performance among all the compared algorithms. Only for UF8 and UF9, MOEA/D-CDE performs significantly worse than MOEA/D- STM.

4.3.2 Performance Comparison on WFG instances

Table 4 shows the performance comparison of all the compared algorithms on the WFG test problems, regarding IGD. Experimental results in Table 4 indicate that MOEA/D-CDE is also competitive in solving the WFG test problems. MOEA/D-CDE obtains the best IGD values on WFG1, WFG3, WFG4, WFG6 and WFG7, while MOEA/D-FRRMAB and MOEA/D-DRA are respectively best on WFG2 and WFG8. MOEA/D-STM performs best on WFG5 and WFG9.

Table 4
Comparative results of all the compared algorithms on the WFG test problems regarding IGD

Problems \ Algorithms		MOEA/D-DE	MOEA/D-DRA	ENS-MOEA/D	MOEA/D-FRRMAB	MOEA/D-STM	MOEA/D-CDE
WFG1	Mean	1.056E-01 -	1.263E-01 -	5.678E-01 -	2.559E-01 -	3.633E-02 -	**2.482E-02**
	Std	1.06E-01	1.25E-01	2.07E-01	2.22E-01	5.09E-02	**6.11E-03**
WFG2	Mean	1.210E-01 -	7.528E-02 ≈	1.764E-01 -	**6.882E-02** +	9.908E-02 -	7.596E-02
	Std	6.93E-02	1.88E-02	8.13E-02	**2.32E-02**	6.09E-02	5.11E-02
WFG3	Mean	1.316E-02 ≈	1.314E-02 ≈	1.341E-02 -	1.316E-02 ≈	1.316E-02 ≈	**1.314E-02**
	Std	3.77E-05	2.82E-05	3.01E-04	3.56E-05	3.25E-05	**2.25E-05**
WFG4	Mean	1.776E-02 -	1.694E-02 -	1.644E-02 -	1.763E-02 -	1.651E-02 -	**1.574E-02**
	Std	2.41E-03	1.74E-03	1.49E-03	1.46E-03	8.87E-04	**4.87E-04**
WFG5	Mean	6.723E-02 -	6.725E-02 ≈	6.761E-02 -	6.721E-02 ≈	**6.710E-02** +	6.714E-02
	Std	1.25E-04	1.18E-04	5.20E-04	1.59E-04	**7.67E-05**	9.54E-05
WFG6	Mean	2.661E-02 -	2.598E-02 -	2.902E-02 -	2.514E-02 -	2.133E-02 -	**1.551E-02**
	Std	1.50E-02	6.99E-03	1.31E-02	1.25E-02	1.11E-02	**4.20E-03**
WFG7	Mean	2.134E-02 -	1.704E-02 -	2.023E-02 -	1.929E-02 -	1.657E-02 -	**1.635E-02**
	Std	1.16E-02	1.39E-03	1.14E-02	9.47E-03	3.21E-04	**2.36E-04**
WFG8	Mean	3.887E-02 +	**2.868E-02** +	3.558E-02 +	3.788E-02 +	4.755E-02 -	3.994E-02
	Std	5.07E-03	**3.71E-03**	5.52E-02	4.84E-03	1.22E-02	4.67E-02
WFG9	Mean	1.513E-02 -	1.570E-02 -	1.638E-02 -	1.600E-02 -	**1.474E-02** ≈	1.480E-02
	Std	3.27E-04	6.21E-04	1.08E-03	1.02E-03	**2.54E-04**	2.38E-04
-/+/≈		7/1/1	5/1/3	8/1/0	5/2/2	6/1/2	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

Table 5
Comparative results of all the compared algorithms on the WFG test problems regarding HV

Problems \ Algorithms		MOEA/D-DE	MOEA/D-DRA	ENS-MOEA/D	MOEA/D-FRRMAB	MOEA/D-STM	MOEA/D-CDE
WFG1	Mean	5.5112 -	6.8664 -	5.9698 -	4.6059 -	7.0776 -	**8.5524**
	Std	1.47E-01	1.61E-01	1.78E-01	1.52E-01	1.51E-01	**8.12E-02**
WFG2	Mean	8.0716 -	8.3403 -	5.8943 -	8.3308 -	8.1708 -	**8.3756**
	Std	3.54E-01	6.58E-02	5.81E-02	1.07E-01	2.55E-01	**2.99E-02**
WFG3	Mean	**7.9531** ≈	7.9526 ≈	7.7063 -	7.9516 ≈	7.9517 ≈	7.9520
	Std	**3.68E-03**	2.09E-03	4.23E-04	7.17E-04	2.57E-03	6.73E-04
WFG4	Mean	5.2590 -	5.3676 -	5.2636 -	5.2300 -	5.2687 -	**5.4940**
	Std	1.48E-01	1.03E-01	8.17E-02	1.23E-01	1.39E-01	**6.27E-02**
WFG5	Mean	5.1658 ≈	5.1660 ≈	4.1341 -	5.1642 -	5.1671 ≈	**5.1691**
	Std	6.66E-04	7.50E-04	3.15E-02	1.34E-03	6.80E-04	**1.45E-02**
WFG6	Mean	5.1764 -	4.9812 -	5.1296 -	5.2245 -	5.4106 -	**5.5996**
	Std	5.14E-01	2.93E-01	6.69E-01	5.67E-01	4.95E-01	**2.33E-01**
WFG7	Mean	5.7808 ≈	**5.8893** +	5.6743 -	5.8225 ≈	5.7796 ≈	5.7693
	Std	3.17E-01	**1.88E-01**	1.63E-01	1.33E-01	2.30E-01	1.35E-01
WFG8	Mean	4.8798 -	**5.1071** +	4.6298 -	4.8987 -	4.7161 -	4.9429
	Std	2.52E-01	**9.03E-02**	1.39E-01	2.05E-01	3.16E-01	1.23E-01
WFG9	Mean	5.6016 -	5.6023 -	5.6012 -	5.5993 -	5.6033 -	**5.6044**
	Std	5.58E-03	1.12E-02	1.49E-02	4.27E-03	1.22E-02	**2.93E-03**
-/+/≈		6/0/3	5/2/2	9/0/0	7/0/2	6/0/3	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

As observed from Table 4, MOEA/D-CDE and MOEA/D-STM perform significantly better than MOEA/D-DE, MOEA/D-DRA, ENS-MOEA/D and MOEA/D-FRRMAB on WFG1. Regarding WFG2, MOEA/D-DE and ENS-MOEA/D cannot solve it very well. For WFG3-WFG9, all the compared algorithms can properly approximate the true *PFs*, as their IGD results are all under the accuracy level of 10^{-2}. The Wilcoxon rank sum test shows that MOEA/D-CDE is similar to MOEA/D-DE on WFG3, to MOEA/D-DRA on WFG2, WFG3 and WFG5, to MOEA/D-FRRMAB on WFG3 and WFG5, and to MOEA/D-STM on WFG3 and WFG9. To conclude, MOEA/D-CDE performs better than or similarly to MOEA/D-DE, MOEA/D-DRA, ENS-MOEA/D, MOEA/D-FRRMAB and MOEA/D-STM on at least 7 out of 9 WFG test problems, which further confirms the advantages of our algorithm. It is worth noting that MOEA/D-CDE and MOEA/D-FRRMAB have shown a comparable performance on most of the WFG test problems. This is mainly because both of them are designed based on the adaptive selection of DE operators. However, our experimental results validate that our proposed algorithm with composite DE operator pools has a superior overall performance.

Table 5 provides comparative results of all the compared algorithms on the WFG test problems regarding HV. It is observed that MOEA/D-CDE performs best on 6 (*i.e.*, WFG1, WFG2, WFG4-WFG6, and WFG9) out of 9 WFG test problems. In more detail, MOEA/D-CDE performs better than or similarly to MOEA/D-DE, ENS-MOEA/D, MOEA/D-FRRMAB and MOEA/D-STM on all the WFG test problems. Regarding the comparison with MOEA/D-DRA, MOEA/D-CDE performs better or similarly on 7 out of 9 WFG test problems. These experimental results with respect to HV also confirm the advantages of MOEA/D-CDE, when compared to the other algorithms in solving all the WFG test problems.

Table 6
Comparative results of all the MOEA/D-CDE variants on the UF test problems regarding IGD

Problems \ Algorithms		CDE-op.1	CDE-op.2	CDE-op.3	CDE-op.4	CDE-op.5	MOEA/D-CDE
UF1	Mean	8.702E-04 -	9.019E-04 -	8.201E-04 ≈	8.266E-04 -	**7.976E-04** +	8.072E-04
	Std	3.15E-05	4.08E-05	4.59E-05	8.84E-05	**1.59E-05**	2.83E-05
UF2	Mean	1.185E-03 -	1.271E-03 -	1.097E-03 -	5.872E-03 -	1.182E-03 -	**9.283E-04**
	Std	4.55E-04	3.03E-04	2.83E-04	1.52E-02	6.83E-04	**1.31E-04**
UF3	Mean	**1.285E-03** +	2.407E-03 -	2.669E-03 -	9.876E-03 -	2.086E-03 -	1.443E-03
	Std	**7.70E-04**	1.84E-03	1.66E-03	1.29E-02	1.90E-03	8.10E-04
UF4	Mean	**2.899E-02** +	4.408E-02 -	5.206E-02 -	5.366E-02 -	3.289E-02 ≈	3.190E-02
	Std	**4.65E-04**	1.46E-03	2.70E-03	3.56E-03	5.94E-04	6.77E-04
UF5	Mean	9.201E-02 ≈	1.988E-01 -	1.671E-01 -	2.519E-01 -	**8.413E-02** +	9.101E-02
	Std	6.43E-02	5.69E-02	6.33E-02	1.08E-01	**1.01E-02**	2.73E-02
UF6	Mean	**5.527E-02** ≈	1.035E-01 -	7.106E-02 -	3.034E-01 -	6.994E-02 -	5.903E-02
	Std	**4.73E-03**	1.34E-01	6.57E-02	1.75E-01	4.76E-02	6.39E-03
UF7	Mean	8.611E-04 -	9.281E-04 -	9.743E-04 -	3.052E-03 -	8.091E-04 ≈	**8.006E-04**
	Std	2.36E-05	3.45E-05	2.30E-04	3.98E-03	1.48E-04	**1.66E-04**
UF8	Mean	3.887E-02 -	3.081E-02 ≈	6.032E-02 -	**2.076E-02** +	5.088E-02 -	2.908E-02
	Std	4.82E-03	1.38E-02	1.43E-02	**2.09E-03**	8.75E-03	2.53E-03
UF9	Mean	3.033E-02 -	5.005E-02 -	4.035E-02 -	4.428E-02 -	4.481E-02 -	**2.154E-02**
	Std	2.90E-02	4.95E-02	4.08E-02	4.84E-02	4.86E-02	**5.66E-02**
UF10	Mean	2.239E-01 ≈	3.997E-01 -	2.112E-01 ≈	3.779E-01 -	2.434E-01 ≈	**2.168E-01**
	Std	5.46E-02	5.25E-02	3.98E-02	6.01E-02	5.93E-02	**4.31E-02**
-/+/≈		5/2/3	9/0/1	8/0/2	9/1/0	5/2/3	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

Table 7
Comparative results of all the compared algorithms on the UF test problems regarding HV

Problems \ Algorithms		CDE-op.1	CDE-op.2	CDE-op.3	CDE-op.4	CDE-op.5	MOEA/D-CDE
UF1	Mean	3.6644 ≈	3.6638 ≈	3.6614 -	3.6619 -	3.6644 ≈	**3.6645**
	Std	1.55E-04	4.66E-04	1.59E-03	1.56E-03	2.42E-04	**8.54E-04**
UF2	Mean	3.6595 -	3.6581 -	3.6602 -	3.6301 -	3.6611 ≈	**3.6643**
	Std	8.27E-03	7.68E-03	5.49E-03	6.20E-02	6.25E-03	**5.42E-03**
UF3	Mean	**3.6643** +	3.6625 ≈	3.6607 -	3.6153 -	3.6630 ≈	3.6636
	Std	**1.14E-03**	2.59E-03	8.18E-03	8.92E-02	3.20E-03	4.57E-03
UF4	Mean	3.2348 -	3.2094 -	3.1829 -	3.1694 -	3.2268 -	**3.2438**
	Std	1.70E-02	7.36E-03	1.11E-02	1.86E-02	2.26E-02	**2.14E-02**
UF5	Mean	3.3198 +	3.2794 ≈	3.1351 -	2.6826 -	**3.3203** +	3.2631
	Std	2.31E-02	7.26E-01	1.47E-01	2.60E-01	**2.04E-02**	3.28E-01
UF6	Mean	3.2255 -	3.1165 -	3.1822 -	2.6161 -	3.2013 -	**3.2465**
	Std	6.05E-02	3.11E-01	2.24E-01	3.12E-01	9.46E-02	**1.38E-01**
UF7	Mean	**3.4977** ≈	3.4957 ≈	3.4943 -	3.4606 -	3.4975 ≈	3.4968
	Std	**1.05E-03**	2.35E-03	4.36E-03	5.07E-02	8.98E-04	3.25E-03
UF8	Mean	7.3866 -	7.3644 -	7.3653 -	**7.4042** +	7.3835 -	7.3927
	Std	1.52E-02	2.59E-02	1.74E-02	**3.44E-03**	1.62E-02	2.02E-02
UF9	Mean	**7.6969** +	7.6112 -	7.6421 -	7.6329 -	7.6361 -	7.6828
	Std	**1.63E-01**	2.26E-01	1.49E-01	2.12E-01	2.13E-01	1.41E-01
UF10	Mean	**4.0851** +	3.7792 -	3.8265 ≈	3.8686 +	3.7591 -	3.8237
	Std	**5.00E-01**	3.01E-01	3.47E-01	3.15E-01	5.38E-01	2.51E-01
-/+/≈		4/4/2	6/0/4	9/0/1	8/2/0	5/1/4	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

4.4 The Effectiveness of the Composite DE Operator Pools

In order to verify the effectiveness of our proposed composite DE operator pools, MOEA/D-CDE is further compared with five MOEA/D-CDE variants, *i.e.*, CDE-op.1, CDE-op.2, CDE-op.3, CDE-op.4, and CDE-op.5. It is noted that CDE-op.*i* (*i*=1, 2, 3, 4) adopts the operator pool op_i defined in Section 3.1, while CDE-op.5 employs all the four DE mutation strategies used in this paper, *i.e.*, "DE/rand/1", "DE/rand/2", "DE/current-to-rand/1" and "DE/rand/2". In order to allow a fair comparison, all the parameters settings for all the MOEA/D-CDE variants are set the same with those of MOEA/D-CDE, as introduced in Section 4.3.

Table 6 provides comparative results of MOEA/D-CDE with respect to all the other MOEA/D-CDE variants adopted here, on the UF test problems, regarding IGD. The experimental results show that MOEA/D-CDE performs best on UF2, UF7, UF9 and UF10, CDE-op.1 is best on UF3, UF4 and UF6, CDE-op.4 gets the best performance on UF8, and CDE-op.5 obtains the best results on UF1 and UF5. The Wilcoxon's rank sum test shows that MOEA/D-CDE performs similarly to CDE-op. 1 on UF5, UF6 and UF10, to CDE-op.2 on UF8, to CDE-op.3 on UF1 and UF10, to CDE-op.5 on UF4, UF7 and UF10. These experimental results reveal that certain DE operator pools can perform best for some UF test problems. However, the overall performance of MOEA/D-CDE is better when considering all the UF test problems, as MOEA/D-CDE is respectively better than CDE-op.1, CDE-op.2, CDE-op.3, CDE-op.4 and CDE-op.5 on 5, 9, 8, 9 and 5 out of 10 UF test problems. On the other hand, CDE-op.1, CDE-op.4 and CDE-op.5 only outperform MOEA/D-CDE on 2, 1 and 2 test problems, respectively. CDE-op.2 and CDE-op.3 are unable to outperform MOEA/D-CDE on any of the UF test problems.

Table 7 further gives comparative results of MOEA/D-CDE on the UF test problems using HV, when compared to the other MOEA/D-CDE variants. As observed from Table 7, MOEA/D-CDE performs best on 4 (*i.e.*, UF1-UF2, UF4, UF6) out of 10 UF test problems. One difference with the above IGD results is that MOEA/D-CDE can't outperform CDE-op.1 when considering this HV metric, but only obtains comparable results with CDE-op.1 as MOEA/D-CDE performs better than, similar to, and worse than CDE-op.1 on 4, 2 and 4 UF test problems. For the remaining MOEA/D-CDE variants, similar conclusions with that of the above IGD results can be made that MOEA/D-CDE performs significantly better as it performs better than or similar to CDE-op.2, CDE-op.3, CDE-op.4 and CDE-op.5 on 10, 10, 8 and 9 out of 10 UF test problems, respectively.

Table 8 provides comparative results of MOEA/D-CDE with respect to all the MOEA/D-CDE variants on the WFG test problems, showing that MOEA/D-CDE is best on WFG8 and WFG9. Furthermore, CDE-op.1 performs best on WFG1, WFG4 and WFG5, CDE-op.3 is best on WFG3, CDE-op.4 obtains the best results on WFG7, and CDE-op.5 gets the best results on WFG2 and WFG6. Regarding the comparison with CDE-op.2, CDE-op.3, and CDE-op.4, MOEA/D-CDE performs significantly better as it is better or similar on 8 out of 9 WFG test problems. For CDE-op.1 and CDE-op5, the Wilcoxon's rank sum test shows that they obtain statistically similar results with MOEA/D-CDE on 6 out of 9 WFG test problems. Besides that, MOEA/D-CDE performs better than CDE-op.1 on WFG6, and CDE-op5 on WFG4 and WFG7. Therefore, MOEA/D-CDE performs slightly better than CDE-op5, but worse than CDE-op.1.

Table 8
Comparative results of all the MOEA/D-CDE variants on the WFG test problems regarding IGD

Problems \ Algorithms		CDE-op.1	CDE-op.2	CDE-op.3	CDE-op.4	CDE-op.5	MOEA/D-CDE
WFG1	Mean	**2.301E-02** +	9.733E-02 -	2.153E-01 -	1.632E-01 -	2. 451E-02 ≈	2.482E-02
	Std	**2.20E-03**	9.10E-02	1.82E-01	2.34E-01	6.02E-03	6.11E-03
WFG2	Mean	7.609E-02 ≈	7.612E-02 ≈	9.052E-02 -	1.517E-01 -	**7.575E-02** ≈	7.596E-02
	Std	1.19E-02	3.91E-02	3.47E-02	8.33E-02	**2.35E-02**	5.11E-02
WFG3	Mean	1.313E-02 ≈	1.320E-02 ≈	**1.280E-02** +	1.297E-02 +	1.315E-02 ≈	1.314E-02
	Std	2.13E-05	7.70E-05	**2.16E-04**	1.92E-04	2.35E-05	2.25E-05
WFG4	Mean	**1.563E-02** +	1.843E-02 -	1.732E-02 -	1.710E-02 -	1.701E-02 -	1.574E-02
	Std	**6.44E-04**	3.32E-03	1.57E-03	1.61E-03	2.28E-03	4.87E-04
WFG5	Mean	**6.710E-02** ≈	6.742E-02 -	6.754E-02 -	6.734E-02 -	6.712E-02 ≈	6.714E-02
	Std	**5.42E-03**	5.17E-03	1.41E-04	1.04E-04	5.52E-03	9.54E-05
WFG6	Mean	1.817E-02 -	1.557E-02 ≈	2.677E-02 -	3.381E-02 -	**1.399E-02** +	1.551E-02
	Std	1.45E-02	1.38E-02	1.14E-02	5.30E-03	**4.86E-05**	4.20E-03
WFG7	Mean	1.636E-02 ≈	1.678E-02 -	1.641E-02 ≈	**1.630E-02** ≈	1.692E-02 -	1.635E-02
	Std	1.87E-04	2.11E-04	9.76E-03	**6.88E-03**	1.96E-04	2.36E-04
WFG8	Mean	4.016E-02 ≈	6.085E-02 -	5.382E-02 -	1.071E-01 -	4.102E-02 ≈	**3.994E-02**
	Std	4.51E-02	7.60E-02	4.64E-02	1.14E-01	4.54E-02	**4.67E-02**
WFG9	Mean	1.482E-02 ≈	1.535E-02 -	1.594E-02 -	1.612E-02 -	1.504E-02 ≈	**1.480E-02**
	Std	1.56E-04	3.41E-04	1.38E-03	1.04E-03	3.57E-04	**2.38E-04**
-/+/≈		1/2/6	6/0/3	7/1/1	7/1/1	2/1/6	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

Table 9
Comparative results of all the MOEA/D-CDE variants on the WFG test problems regarding HV

Problems \ Algorithms		CDE-op.1	CDE-op.2	CDE-op.3	CDE-op.4	CDE-op.5	MOEA/D-CDE
WFG1	Mean	**8.8999** +	7.5038 -	6.6769 -	6.9624 -	8.4248 ≈	8.5524
	Std	**5.92E-01**	1.62E-01	1.83E-01	1.57E-01	9.54E-01	8.12E-02
WFG2	Mean	8.3602 ≈	8.2947 -	8.2158 -	8.0046 -	8.3735 ≈	**8.3756**
	Std	1.818E-02	1.76E-01	1.44E-01	3.34E-01	2.81E-02	**2.99E-02**
WFG3	Mean	7.9518 ≈	7.9511 ≈	**7.9547** +	7.9546 +	7.9519 ≈	7.9520
	Std	9.437E-04	2.85E-03	**5.94E-03**	5.86E-03	7.80E-04	6.73E-04
WFG4	Mean	**5.5599** +	5.2197 -	5.2581 -	5.2328 -	5.4023 -	5.4940
	Std	**4.511E-02**	1.45E-01	1.60E-01	1.30E-01	6.86E-02	6.27E-02
WFG5	Mean	5.1690 ≈	5.1673 -	5.1647 -	5.1648 -	5.1681 ≈	**5.1691**
	Std	1.325E-02	1.34E-02	1.61E-03	1.30E-03	1.23E-02	**1.45E-02**
WFG6	Mean	5.5160 -	5.6063 ≈	5.0673 -	4.6939 -	**5.6751** +	5.5996
	Std	4.562E-01	3.74E-01	5.04E-01	1.38E-01	**2.62E-04**	2.33E-01
WFG7	Mean	5.8581 +	5.7036 -	5.9019 +	**5.9199** +	5.6676 -	5.7693
	Std	2.700E-01	8.94E-02	1.05E-01	**7.28E-02**	2.23E-01	1.35E-01
WFG8	Mean	**4.9775** +	4.9306 -	4.6732 -	4.6732 -	4.9745 +	4.9429
	Std	**1.919E-01**	2.07E-01	2.31E-01	3.12E-01	1.32E-01	1.23E-01
WFG9	Mean	5.6040 ≈	5.5998 -	5.6020 ≈	5.5780 -	5.6022 ≈	**5.6044**
	Std	1.185E-03	2.46E-03	2.51E-02	2.61E-01	1.85E-03	**2.93E-03**
-/+/≈		1/4/4	7/0/2	6/2/1	7/2/0	2/2/5	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

Table 9 further gives comparative results of MOEA/D-CDE on the WFG test problems using HV,

when compared to the other MOEA/D-CDE variants. As observed from Table 9, MOEA/D-CDE performs best on WFG2, WFG5 and WFG 9. One difference with the above IGD results is that MOEA/D-CDE only obtains comparable results with CDE-op.5 as MOEA/D-CDE performs better than, similar to, and worse than CDE-op.5 on 2, 5 and 2 WFG test problems, respectively. For the remaining MOEA/D-CDE variants, similar conclusions with that of the above IGD results can be observed. MOEA/D-CDE performs significantly better than CDE-op.2, CDE-op.3, and CDE-op.4, but worse than CDE-op.1.

Based on the comparison summary in the last rows of Tables 6-9, the effectiveness of the composite DE operator pools used in MOEA/D-CDE is experimentally validated (*i.e.*, it is shown that the use of composite DE operator pools enhances the overall performance of our proposed approach when considering all the UF and WFG test problems).

4.5 The Effectiveness of Adaptive Parameter Control Strategy

To investigate the effectiveness of the adaptive parameter control strategy adopted in the composite DE operator pools, our algorithm is further compared to the MOEA/D-CDE variants with fixed parameters settings and other adaptive parameters settings. In MOEA/D-CDE, only the value of CR is fixed while the value of F is adaptively determined as introduced in Section 3.3. The compared MOEA/D-CDE variants follow the same procedures as MOEA/D-CDE except that the composite DE operator pools use different control parameters settings, which are described as follows.

(1) CDE-1: the parameters settings for the DE operator pools are set to $F = 0.1$ and $CR = 0.1$;

(2) CDE-2: the parameters settings for the DE operator pools are set to $F = 1.0$ and $CR = 0.9$;

(3) CDE-3: the parameters settings for the DE operator pools are set to $F = 0.8$ and $CR = 0.2$;

(4) CDE-4: the parameter CR for the DE operator pools is adaptively determined as introduced in [20], and F is set to the fixed value of 1.0;

(5) CDE-5: the settings of F and CR for the DE operator pools are all adaptively determined as introduced in Section 3.3.

Tables 10-13 respectively give the experimental results of all the compared algorithms on the UF and the WFG test problems using IGD and HV. As observed from Table 10, our algorithm performs best on 6 out of 10 UF test problems, *i.e.*, UF1-UF3 and UF6-UF9. Moreover, our algorithm respectively outperforms CDE-1, CDE-2, CDE-3, CDE-4 and CDE-5 on 8, 10, 8, 8 and 7 out of 10 UF test problems. Nevertheless, MOEA/D-CDE is only beaten by CDE-1 on UF4 and UF10, by CDE-3 on UF10, by CDE-op.4 on UF10, and by CDE-5 on UF4-UF5 and UF10. The Wilcoxon's rank sum test also indicates that our algorithm performs similarly to CDE-3 on UF4 and to CDE-4 on UF4. Therefore, when considering all the UF test problems on IGD, our algorithm is better than all the MOEA-D/CDE variants with other parameters settings. Moreover, the HV results of all the compared algorithms in Table 11 further confirm the effectiveness of our adaptive parameter control strategy in solving the UF test problems, as our algorithm performs better than or similar to CDE-1, CDE-2, CDE-3, CDE-4 and CDE-5 on 9, 9, 8, 9 and 8 out of 10 UF test problems.

Table 10
Comparative results of the MOEA/D-CDE variants using different parameters settings on the UF test problems regarding IGD

Problems \ Algorithms		CDE-1	CDE-2	CDE-3	CDE-4	CDE-5	MOEA/D-CDE
UF1	Mean	1.511E-03 -	1.917E-03 -	2.600E-03 -	4.474E-03 -	1.816E-03 -	**8.072E-04**
	Std	1.21E-02	1.21E-04	6.26E-04	2.20E-04	4.48E-04	**2.83E-05**
UF2	Mean	3.678E-03 -	2.472E-03 -	2.300E-03 -	2.819E-03 -	2.337E-03 -	**9.283E-04**
	Std	1.74E-03	3.29E-04	1.06E-03	1.03E-03	2.50E-03	**1.31E-04**
UF3	Mean	1.732E-02 -	7.549E-03 -	5.380E-02 -	6.073E-02 -	1.722E-03 -	**1.443E-03**
	Std	3.78E-02	4.81E-03	1.96E-02	8.90E-03	1.64E-02	**8.10E-04**
UF4	Mean	**2.631E-02** +	3.557E-02 -	3.210E-02 ≈	2.921E-02 ≈	2.743E-02 +	3.190E-02
	Std	**1.46E-03**	2.61E-04	5.17E-04	5.27E-04	9.93E-04	6.77E-04
UF5	Mean	2.037E-01 -	1.008E-01 -	1.331E-01 -	1.370E-01 -	**8.021E-02** +	9.101E-02
	Std	7.77E-02	1.96E-02	1.54E-02	2.22E-02	**4.11E-03**	2.73E-02
UF6	Mean	2.395E-01 -	6.334E-02 -	6.980E-02 -	6.537E-02 -	6.077E-02 -	**5.903E-02**
	Std	1.67E-01	6.66E-03	3.18E-02	1.98E-03	4.09E-03	**6.39E-03**
UF7	Mean	6.881E-02 -	2.062E-03 -	2.700E-03 -	4.125E-03 -	2.596E-03 -	**8.006E-04**
	Std	1.21E-01	1.44E-04	3.99E-04	6.05E-04	2.69E-04	**1.66E-04**
UF8	Mean	4.713E-02 -	3.498E-02 -	3.590E-02 -	3.898E-02 -	4.156E-02 -	**2.908E-02**
	Std	2.13E-02	4.62E-03	5.62E-03	2.82E-03	8.66E-03	**2.53E-03**
UF9	Mean	6.311E-02 -	4.303E-02 -	4.000E-02 -	4.923E-02 -	4.475E-02 -	**2.154E-02**
	Std	4.39E-02	1.96E-02	5.28E-03	2.81E-03	3.47E-02	**5.66E-02**
UF10	Mean	1.795E-01 +	3.852E-01 -	1.769E-01 +	**1.650E-01** +	1.853E-01 +	2.168E-01
	Std	3.48E-02	8.17E-02	2.14E-02	**3.23E-02**	2.58E-02	4.31E-02
-/+/≈		8/2/0	10/0/0	8/1/1	8/1/1	7/3/0	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

Table 11
Comparative results of the MOEA/D-CDE variants using different parameters settings on the UF test problems regarding HV

Problems \ Algorithms		CDE-1	CDE-2	CDE-3	CDE-4	CDE-5	MOEA/D-CDE
UF1	Mean	3.6005 -	3.6606 -	3.6616 -	3.6585 -	3.6631 ≈	**3.6645**
	Std	7.64E-02	6.07E-04	1.26E-03	7.43E-04	1.41E-03	**8.54E-04**
UF2	Mean	3.6466 -	3.6558 -	3.6521 -	3.6576 -	3.6512 -	**3.6643**
	Std	1.74E-02	6.76E-03	1.56E-02	1.12E-02	2.17E-02	**5.42E-03**
UF3	Mean	2.8714 -	3.6532 -	3.4821 -	3.4907 -	3.6241 ≈	**3.6636**
	Std	1.39E-01	1.03E-02	9.70E-02	5.98E-02	5.82E-02	**4.57E-03**
UF4	Mean	**3.2495** ≈	3.2302 -	3.2163 -	3.2206 -	3.2338 -	3.2438
	Std	**2.51E-02**	1.01E-02	2.89E-02	2.66E-02	2.77E-02	2.14E-02
UF5	Mean	2.7593 -	3.2684 ≈	3.0032 -	3.1719 -	**3.3122** +	3.2631
	Std	3.11E-01	7.25E-02	1.22E-01	1.51E-01	**3.53E-02**	3.28E-01
UF6	Mean	2.7693 -	3.2166 ≈	3.0877 -	3.1462 -	3.1947 -	**3.2465**
	Std	3.35E-01	2.11E-02	1.19E-01	1.03E-01	7.92E-02	**1.38E-01**
UF7	Mean	3.2831 -	3.4938 ≈	3.4946 ≈	3.4921 -	3.4943 -	**3.4968**
	Std	3.70E-01	1.20E-03	1.25E-03	1.20E-03	1.07E-03	**3.25E-03**
UF8	Mean	7.3082 -	7.2947 -	7.3214 -	7.3123 -	7.3536 -	**7.3927**
	Std	1.70E-01	7.03E-03	1.40E-02	7.13E-03	1.82E-02	**2.02E-02**
UF9	Mean	7.5286 -	7.6461 -	**7.6982** +	7.6741 -	7.6521 -	7.6828
	Std	2.16E-01	9.18E-02	**2.51E-02**	2.28E-02	1.73E-01	1.41E-01
UF10	Mean	6.2047 +	4.7846 +	6.3321 +	**6.4348** +	6.2039 +	3.8237
	Std	5.50E-01	4.89E-01	1.61E-01	**3.75E-01**	4.07E-01	2.51E-01
-/+/≈		8/1/1	6/1/3	7/2/1	9/1/0	6/2/2	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

Table 12
Comparative results of the MOEA/D-CDE variants using different parameters settings on the WFG test problems regarding IGD

Problems \ Algorithms		CDE-1	CDE-2	CDE-3	CDE-4	CDE-5	MOEA/D-CDE
WFG1	Mean	2.486E-02 ≈	5.215E-02 -	2.543E-02 ≈	6.847E-02 -	**2.264E-02** +	2.482E-02
	Std	1.06E-01	9.29E-02	9.11E-02	1.41E-01	**2.31E-04**	6.11E-03
WFG2	Mean	2.448E-01 -	**3.878E-02** +	1.038E-01 -	7.718E-02 ≈	8.375E-02 -	7.596E-02
	Std	8.58E-04	**8.61E-03**	5.71E-02	2.33E-02	1.41E-04	5.11E-02
WFG3	Mean	1.321E-02 ≈	1.329E-02 -	1.316E-02 ≈	1.315E-02 ≈	1.319E-02 ≈	**1.314E-02**
	Std	1.52E-04	6.72E-05	4.89E-05	4.69E-05	7.42E-05	**2.25E-05**
WFG4	Mean	**1.551E-02** +	2.227E-02 -	1.609E-02 -	1.569E-02 +	1.587E-02 ≈	1.574E-02
	Std	**4.80E-04**	3.98E-03	5.11E-04	6.87E-04	4.62E-04	4.87E-04
WFG5	Mean	6.698E-02 ≈	6.687E-02 ≈	**6.612E-02** +	6.718E-02 ≈	6.700E-02 ≈	6.714E-02
	Std	5.56E-03	5.92E-03	**7.55E-03**	3.21E-03	5.45E-03	9.54E-05
WFG6	Mean	2.726E-02 -	**1.412E-02** +	2.618E-02 -	1.906E-02 -	1.536E-02 ≈	1.551E-02
	Std	8.53E-03	**8.59E-05**	1.59E-02	1.93E-02	5.70E-03	4.20E-03
WFG7	Mean	**1.631E-02** ≈	1.683E-02 -	1.633E-02 ≈	1.680E-02 -	1.671E-02 -	1.635E-02
	Std	**8.57E-04**	2.29E-05	4.06E-03	3.14E-03	1.38E-04	2.36E-04
WFG8	Mean	6.964E-02 -	4.559E-02 -	4.784E-02 -	4.772E-02 -	5.452E-02 -	**3.994E-02**
	Std	4.05E-02	5.36E-03	4.32E-03	4.82E-03	4.36E-02	**4.67E-02**
WFG9	Mean	1.486E-02 ≈	1.702E-02 -	1.488E-02 ≈	1.481E-02 ≈	1.483E-02 ≈	**1.480E-02**
	Std	8.94E-04	1.20E-03	7.33E-04	1.26E-04	1.51E-04	**2.38E-04**
-/+/≈		3/1/5	6/2/1	4/1/4	4/1/4	3/1/5	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

Table 13
Comparative results of the MOEA/D-CDE variants using different parameters settings on the WFG test problems regarding HV

Problems \ Algorithms		CDE-1	CDE-2	CDE-3	CDE-4	CDE-5	MOEA/D-CDE
WFG1	Mean	8.7402 +	8.2016 -	9.0058 +	8.4113 -	**9.0455** +	8.5524
	Std	7.35E-01	1.01E+00	2.93E-01	4.57E-01	**6.55E-02**	8.12E-02
WFG2	Mean	7.8163 -	**8.3885** +	8.2961 -	8.3649 ≈	8.3528 -	8.3756
	Std	3.59E-02	**5.69E-02**	1.86E-01	1.40E-02	1.48E-02	2.99E-02
WFG3	Mean	7.9512 ≈	7.9501 -	7.9518 ≈	7.9518 ≈	7.9520 ≈	**7.9520**
	Std	8.57E-04	1.48E-03	4.17E-04	4.32E-04	4.69E-04	**6.73E-04**
WFG4	Mean	**5.6239** +	5.0949 -	5.4484 -	5.5873 +	5.6096 +	5.4941
	Std	**2.09E-02**	1.45E-01	4.64E-02	3.19E-02	2.61E-02	6.27E-02
WFG5	Mean	5.1696 ≈	5.1653 ≈	5.1706 ≈	5.1687 ≈	**5.1707** ≈	5.1691
	Std	1.22E-02	1.31E-02	1.67E-02	9.30E-03	**1.28E-02**	1.45E-02
WFG6	Mean	4.8897 -	**5.6731** +	5.0745 -	5.5389 -	5.6081 ≈	5.5996
	Std	2.75E-01	**6.43E-04**	5.35E-01	5.20E-01	2.64E-01	2.33E-01
WFG7	Mean	**5.9101** +	5.6736 -	5.8005 +	5.6743 -	5.6703 -	5.7693
	Std	**2.52E-01**	3.39E-04	2.66E-01	7.85E-03	4.44E-01	1.35E-01
WFG8	Mean	4.7221 -	4.9414 ≈	4.9132 -	4.9397 -	4.8268 -	**4.9429**
	Std	2.39E-01	2.22E-01	1.42E-01	1.37E-01	1.44E-01	**1.23E-01**
WFG9	Mean	5.5915 ≈	5.5897 -	5.6042 ≈	5.6041 ≈	5.6042 ≈	**5.6044**
	Std	9.78E-02	6.13E-03	1.87E-03	1.35E-03	1.05E-03	**2.93E-03**
-/+/≈		3/3/3	5/2/2	4/2/3	4/1/4	3/2/4	

"-", "+" and "≈" respectively denote the performance of the corresponding algorithm is worse than, better than, and similar to that of MOEA/D-CDE, according to Wilcoxon's rank sum test at a 0.05 significance level

From Table 12, it is observed that our algorithm obtains the best results on WFG3, WFG8 and WFG9. The Wilcoxon's rank sum test shows that our algorithm performs similarly to CDE-1, CDE-2, CDE-3, CDE-4 and CDE-5 on 5, 1, 4, 4, 5 out of 9 WFG test problems, which indicates that the performance of our algorithm is insensitive to the parameters settings of F and CR when solving most of the WFG test problems. The final results in the last row of Table 12 show that our algorithm outperforms CDE-1, CDE-2, CDE-3, CDE-4 and CDE-5 on 3, 6, 4, 4 and 3 WFG test problems, but only underperforms CDE-1, CDE-2, CDE-3, CDE-4 and CDE-5 on 1, 2, 1, 1 and 1 WFG test problems, respectively. Therefore, our algorithm still performs better when considering all the WFG test problems. Table 13 further provides comparative results of MOEA/D-CDE on the WFG test problems using HV when compared to all the MOEA/D-CDE variants. As observed from Table 13, our algorithm also achieves the best results on WFG3, WFG8 and WFG9. one difference with the above IGD results is that MOEA/D-CDE only obtains comparable results with CDE-1 as MOEA/D-CDE performs better than, similar to, and worse than CDE-1 on 3, 3 and 3 WFG test problems, respectively. For the remaining MOEA/D-CDE variants (*i.e.*, CDE-2, CDE-3, CDE-4 and CDE-5), similar conclusions with that of the above IGD results can be found that MOEA/D-CDE performs better than CDE-2, CDE-3, CDE-4, and CDE-5. Therefore, these experimental results on WFG test problems further confirm the effectiveness of the adaptive parameter control strategy adopted in MOEA-D/CDE.

4.6 Time Complexity Analysis

In this subsection, the worst time complexity analysis of MOEA/D-CDE is provided and compared to that of other algorithms. Based on the pseudo-code of MOEA/D-CDE in **Algorithm 4**, the worst time complexity of MOEA/D-CDE is mainly determined by the evolutionary loop in lines 5-47 of **Algorithm 4**. It is noted that when calculating the worst time complexity, the number of decision variables n and the number of objectives m are ignored as they are much smaller than the population size N. In line 5, the worst time complexity to perform 10-tournament selection is $O(N)$. For the evolutionary loop in lines 6-42, it will take the worst time complexity which is $O(W \times N)$ to select an operator pool in lines 8-12, and is $O(N)$ to choose a parent set in lines 13-17; the new offspring are generated using the new scaling factor F in lines 18-19 with the worst time complexity $O(N)$. Then, the reference point is updated in lines 23-27 with the worst time complexity $O(m \times N)$, and at most n_r parents in set E are replaced in lines 28-37 with the worst time complexity $O(N^2)$ as the size of E may be N with a probability $(1-\delta)$. In lines 39-40, the worst time complexity to update the performance array $Array(2)(W)$ is $O(N)$. At last, the F', g and the utility values π_i (i=1, 2,..., N) are renewed in lines 43-47 with the worst time complexity $O(N)$. In summary, the worst time complexity of MOEA/D-CDE can be simplified to $O(N^2)$, which has a worst time complexity comparable with that of MOEA/D-DE [36], MOEA/D-DRA [55], and MOEA/D-FRRMAB [29], while the worst time complexity of MOEA/D-STM is higher due to the use of a stable matching model [28].

To further study the extra computational burden induced by the adaptive composite operator selection and parameter control strategy, we have collected the actual CPU time cost of MOEA/D-CDE, MOEA/D-DE, MOEA/D-DRA, MOEA/D-STM and MOEA/D-FRRMAB on UF1-UF10. Their average CPU times from 30 independent runs are listed in Table 14. It is noted that the lowest CPU

time is highlighted in **boldface**. The CPU time required by ENS-MOEA/D is not reported because it is implemented in MATLAB, which makes it very slow [29]. From Table 14, it is observed that MOEA/D-DE was the fastest in solving all the UF problems, while MOEA/D-DRA ranked second in terms of running speed as the dynamic resource assignment strategy designed in MOEA/D-DRA requires some extra computational cost. As MOEA/D-CDE and MOEA/D-FRRMAB are all designed by embedding the adaptive operator selection mechanism into the framework of MOEA/D-DRA, it is certain that they will have a higher computational cost than MOEA/D-DRA, as justified by our experimental results in Table 14. It is also observed that MOEA/D-CDE performed faster than MOEA/D-FRRMAB on all the UF test problems. For MOEA/D-CDE, it is worth consuming an additional 27% of CPU time with respect to MOEA/D-DRA on average for obtaining the performance improvement caused by the adaptive composite operator selection and parameter control strategy, as listed in Table 2. Moreover, it is worth noting that the CPU time of MOEA/D-STM is much longer than that of other MOEA/D variants. That is mainly due to the fact that the stable matching model is time-consuming in the selection process and the source code may not be fully optimized by the authors.

Table 14
Average CPU time (in seconds) cost by all the compared algorithms on the UF test problems

Problems \ Algorithms	MOEA/D-DE	MOEA/D-DRA	MOEA/D-STM	MOEA/D-FRRMAB	MOEA/D-CDE
UF1	**4.432**	5.942	233	8.489	7.987
UF2	**4.764**	6.544	224	8.520	8.072
UF3	**5.314**	6.401	234	8.673	8.358
UF4	**4.656**	6.057	224	8.413	7.583
UF5	**4.713**	5.867	229	8.454	7.746
UF6	**4.927**	6.095	228	8.522	7.856
UF7	**3.972**	5.683	234	7.855	7.217
UF8	**6.476**	9.419	419	12.588	11.268
UF9	**5.754**	8.488	421	12.469	11.545
UF10	**6.397**	9.351	456	13.193	11.453
Average	4.764	6.441	263.909	8.925	8.190

5. Conclusions

In this paper, an adaptive composite operator selection and parameter control strategy for MOEA/D namely MOEA/D-CDE, was proposed. Four DE mutation strategies were used to build the composite operator pools, which can address the limitations of using a single DE mutation strategy when solving different types of MOPs. To adaptively determine the preferred DE operator pool during the evolutionary search, an adaptive composite operator selection mechanism was designed based on the previous search experience. Moreover, some parameters used in the composite operator pools were also automatically determined by using an adaptive parameter control scheme, which was shown to further enhance optimization performance. When embedding the proposed approach into the baseline algorithm MOEA/D-DRA, the optimization performance was substantially improved. Our experimental results validated that MOEA/D-CDE is able to outperform MOEA/D-DE, ENS-MOEA/D, MOEA/D-FRRMAB and MOEA/D-STM on most of the test problems adopted. The effectiveness of

the proposed adaptive composite operator selection and parameter control strategy was also experimentally studied.

As part of our future work, we intend to extend our study to other nature-inspired algorithms for tackling more difficult optimization problems modeled from real-life engineering applications. It is interesting to study how to combine different evolutionary operators from EAs, DE, particle swarm optimization and ant colony optimization into our composite operator pools, so that each composite operator pool can be more effective and can overcome the natural shortcomings of each other.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grants 61402291 and 61170283, National High-Technology Research and Development Program ("863" Program) of China under Grant 2013AA01A212, Ministry of Education in the New Century Excellent Talents Support Program under Grant NCET-12-0649, Foundation for Distinguished Young Talents in Higher Education of Guangdong under Grant 2014KQNCX129, Shenzhen Technology Plan under Grant JCYJ20140418095735608, and Natural Science Foundation of SZU under Grant 201531. It was also supported by CONACyT grant no. 221551. The authors are grateful to both the editor and anonymous reviewers for their constructive comments, which greatly improved the quality of this paper.

Reference:

[1] M. Asafuddoula, T. Ray, R. Sarker, A decomposition based evolutionary algorithm for many objective optimization, IEEE Trans. Evol. Comput. 19 (3) (2014) 445-460.

[2] P.A.N. Bosman, D. Thierens, The balance between proximity and diversity in multiobjective evolutionary algorithms, IEEE Trans. Evol. Comput. 7(2) (2003) 174-188.

[3] S. Biswas, S. Kundu, S. Das, Inducing niching behavior in differential evolution through local information sharing, IEEE Trans. Evol. Comput. 19 (2) (2014) 246-263.

[4] C. Chen, L. Y. Tseng, An improved version of the multiple trajectory search for real value multi-objective optimization problems, Engineering Optimization 46 (10) (2014) 1430-1445.

[5] J.Y. Chen, Q.Z. Lin, Z. Ji, Chaos-based multiobjective immune algorithm with a fine-grained selection mechanism, Soft Comput. 15 (2011) 1273-1288.

[6] C. A. Coello Coello, G. B. Lamont, D. A. Van Veldhuizen, Evolutionary algorithms for solving multi-objective problems, Genetic and Evol. Comput. Springer, 2007.

[7] C. Dai, Y. Wang, A new multiobjective evolutionary algorithm based on decomposition of the objective space for multiobjective optimization, Journal of Applied Mathematics, vol. 2014, 2014, Article ID 906147, 9 pages.

[8] C. Dai, Y.P. Wang, M. Ye, A new multi-objective particle swarm optimization algorithm based on decomposition, Inf. Sci. 325 (2015) 541-557.

[9] K. Deb, M. Goyal, A combined genetic adaptive search (GeneAS) for engineering design, Comput. Inf. Sci. 26 (1996) 30-45.

[10] K. Deb, Multi-objective optimization using evolutionary algorithms, John Wiley & Sons Ltd., New York, 2001.

[11] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182-197.
[12] K. Deb, 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. 19 (4) (2014) 577-601.
[13] M. Ehrgott, Multicriteria optimization, Springer-Verlag, Berlin, Heidelberg, 2006.
[14] S. Ghosh, D. Saurav, S. Roy, A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization, Inf. Sci. 182 (1) (2012) 199-219.
[15] I. Giagkiozis, R. C. Purshouse, P. J. Fleming, Generalized decomposition and cross entropy methods for many-objective optimization, Inf. Sci. 282 (2014) 363-387.
[16] I. Giagkiozis, P. J. Fleming, Methods for multi-objective optimization: An analysis, Inf. Sci. 293 (2015) 338-350.
[17] F. Gu, H. Liu, A novel weight design in multi-objective evolutionary algorithm. In International Conference on Computational Intelligence and Security, 2010, pp. 137-141,.
[18] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (5) (2006) 477-506.
[19] A. W. Iorio, X. Li, Solving rotated multi-objective optimization problems using differential evolution, AI 2004 Advances in Artificial Intelligence, vol. 3339 of the series Lecture Notes in Computer Science, 2005, pp. 861-872.
[20] S. M. Islam, S. Das, S. Ghosh, S. Roy, P. N. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Trans. Systems, Man, and Cyber. Part B: Cyber. 42 (2) (2012) 482-500.
[21] H. Jain, K. Deb, An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: handling constraints and extending to an adaptive approach, IEEE Trans. Evol. Comput. 18 (4) (2014) 602-622.
[22] L. Jiao, H. Wang, R. Shang, F. Liu, A co-evolutionary multi-objective optimization algorithm based on direction vectors, Inf. Sci. 228 (2013) 90-112.
[23] S.W. Jiang, Z.H. Cai, J. Zhang, Y.-S. Ong, Multiobjective optimization by decomposition with Pareto-adaptive weight vectors Pareto-adaptive weight vectors, In International Conference on Natural Computation, 2011, pp. 1260-1264.
[24] S.W. Jiang, Y.-S. Ong, J. Zhang, L. Feng, Consistencies and Contradictions of Performance Metrics in Multiobjective Optimization, IEEE Trans. Cyber. 44(12) (2014) 2391-2404.
[25] D. Kovačević, N. Mladenović, B. Petrović, P. Milošević, DE-VNS: Self-adaptive differential evolution with crossover neighborhood search for continuous global optimization, Comput. & Opera. Research 52 (2014) 157-169.
[26] H. Li, D. Landa-Silva, An adaptive evolutionary multi-objective approach based on simulated annealing, IEEE Trans. Evol. Comput. 19(4) (2011) 561-595.
[27] K. Li, S. Kwong, J. Cao, M.Q. Li, J. Zheng, R. Shen, Achieving balance between proximity and diversity in multi-objective evolutionary algorithm, Inf. Sci. 182 (1) (2012) 220-242.
[28] K. Li, Q.F. Zhang, S. Kwong, M. Li, R. Wang, Stable matching based selection in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 18 (6) (2013) 909-923.
[29] K. Li, A. Fialho, S. Kwong, Q.F. Zhang, Adaptive operator selection with bandits for a

multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 18 (2) (2014) 114-130.

[30] K. Li, S. Kwong, Q.F. Zhang, K. Deb, Inter-relationship based selection for decomposition multiobjective optimization, IEEE Trans. Cyber. 45 (10) (2015) 2076-2088.

[31] M.Q. Li, S. Yang, K. Li, X. Liu, Evolutionary algorithms with segment-based search for multiobjective optimization problems, IEEE Trans. Cyber. 44 (8) (2014) 1295-1313.

[32] Z.P. Liang, R.Z. Song, Q.Z. Lin, Z.H. Du, J.Y. Chen, Z. Ming, J.P. Yu, A double-module immune algorithm for multi-objective optimization problems, Appl. Soft Comput. 35 (2015) 161-174.

[33] Q.Z. Lin, J.Y. Chen, A novel micro-population immune multiobjective optimization algorithm, Comput. & Opera. Research 40 (2013) 1590-1601.

[34] Q.Z. Lin, Q.L. Zhu, P.Z. Huang, J.Y. Chen, Z. Ming, J.P. Yu, A novel hybrid multi-objective immune algorithm with adaptive differential evolution, Comput. & Opera. Research 62 (2015) 95-111.

[35] H. Liu, F. Gu, Y. Cheung, T-MOEA/D: MOEA/D with objective trans-form in multi-objective problems. In International Conference of Information Science and Management Engineering, 2010, pp. 282-285.

[36] H. Liu, Q.F. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Trans. Evol. Comput. 13 (2) (2009) 284-302.

[37] X. Ma, F. Liu, Y. Qi, M. Gong, M. Yin, L. Li, L. Jiao, J. Wu, MOEA/D with opposition-based learning for multiobjective optimization problem, Neurocomputing 146 (2014) 48-64.

[38] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679-1696.

[39] W. K. Mashwani, A. Salhi, A decomposition-based hybrid multiobjective evolutionary algorithm with dynamic resource allocation, Appl. Soft Comput. 12 (9) (2012) 2765-2780.

[40] K. Miettinen, Nonlinear multiobjective optimization, Kluwer Academic Publishers, Boston, 1999.

[41] T. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, J. Wu, MOEA/D with adaptive weight adjustment, IEEE Trans. Evol. Comput. 22(2) (2014) 231-264.

[42] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398-417.

[43] R. Shang, L. Jiao, F. Liu, W. Ma, A Novel Immune Clonal Algorithm for MO Problems, IEEE Trans. Evol. Comput. 16 (1) (2010) 35-50.

[44] R. Storn, K. Price, Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, Berkeley, ICSI, 1995, Available online: http://www1.icsi.berkeley.edu/~storn/litera.html.

[45] Y. Y. Tan, Y. C. Jiao, H. Li, X. Wang, A modification to MOEA/D-DE for multiobjective optimization problems with complicated Pareto sets, Inf. Sci. 213 (2012) 14-38.

[46] L. Tang, X. Wang, A hybrid multiobjective evolutionary algorithm for multiobjective optimization problems, IEEE Trans. Evol. Comput. 17 (1) (2013) 20-45.

[47] S. M. Venske, R. A. Gonçalves, M. R. Delgado, ADEMO/D: Multiobjective optimization by an adaptive differential evolution algorithm, Neurocomputing, 127 (2) (2014) 65-77.

[48] H. Wang, L. Jiao, R. Shang, S. He, F. Liu, A Memetic Optimization Strategy Based on Dimension Reduction in Decision Space, IEEE Trans. Evol. Comput. 23(1) (2015) 69-100.

[49] Y. Wang, Z.X. Cai, Q.F. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2) (2011) 55-66.
[50] G. G. Yen, H. Lu, Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation, IEEE Trans. Evol. Comput. 7 (3) (2003) 253-274.
[51] Z.H. Zhan, J.J. Li, J.N. Cao, J. Zhang, H.S.H. Chung, Y.H. Shi, Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems, IEEE Trans. Cyber. 43 (2013) 445-463.
[52] J. Zhang, A. C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. comput. 13 (5) (2009) 945-958.
[53] Q.F. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712-731.
[54] Q.F. Zhang, H. Li, D. Maringer, E. Tsang, MOEA/D with NBI-style Tchebycheff approach for portfolio management, In IEEE Congress on Evolutionary Computation, 2010, pp. 1-8.
[55] Q.F. Zhang, W. Liu, H. Li, The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances, in Proceedings of the 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 203-208.
[56] Q.F. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization test instances for the CEC 2009 special session and competition, University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, Technical report, 2008, pp. 1-30.
[57] S. Zhao, P. N. Suganthan, Q.F. Zhang, Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Trans. Evol. Comput. 16 (3) (2012) 442-446.
[58] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257-271.
[59] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. Da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117-132.
[60] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Technical report 103, 2001.