Distributed Individuals for Multiple Peaks: A Novel Differential Evolution for Multimodal Optimization Problems

Zong-Gan Chen, *Student Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, Hua Wang, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Locating more peaks and refining the solution accuracy on the found peaks are two challenging issues in solving multimodal optimization problems (MMOPs). To deal with these two challenges, a distributed individuals differential evolution (DIDE) algorithm is proposed in this article based on a distributed individuals for multiple peaks (DIMP) framework and two novel mechanisms. First, the DIMP framework provides sufficient diversity by letting each individual act as a distributed unit to track a peak. Based on the DIMP framework, each individual uses a virtual population controlled by an adaptive range adjustment strategy to explore the search space sufficiently for locating a peak and then gradually approach it. Second, the two novel mechanisms named lifetime mechanism and elite learning mechanism (ELM) cooperate with the DIMP framework. The lifetime mechanism is inspired by the natural phenomenon that every organism will gradually age and has a limited lifespan. When an individual runs out of its lifetime and also has good fitness, it is regarded as an elite solution and will be added to an archive. Then the individual restarts a new lifetime, so as to bring further diversity to locate more peaks. The ELM is proposed to refine the accuracy of those elite solutions in the archive, being efficient in dealing with the solution accuracy issue on the found peaks. The experimental results on 20 multimodal benchmark test functions show that the proposed DIDE algorithm has generally better or competitive performance compared with the state-of-the-art multimodal optimization algorithms.**

***Index Terms*—Differential evolution (DE), distributed individuals DE (DIDE), lifetime mechanism, multimodal optimization.**

Manuscript received January 28, 2019; revised July 30, 2019; accepted September 17, 2019. Date of publication October 1, 2019; date of current version July 30, 2020. This work was supported in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundations of China under Grant 61772207 and Grant 61873097, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform under Grant 2018B050502006. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

Z.-G. Chen and Z.-H. Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

H. Wang is with the College of Engineering and Science, Institute for Sustainable Industries and Liveable Cities, Victoria University, Melbourne, VIC 8001, Australia.

J. Zhang is with the Victoria University, Melbourne, VIC 8001, Australia (e-mail: junzhang@ieee.org).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

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

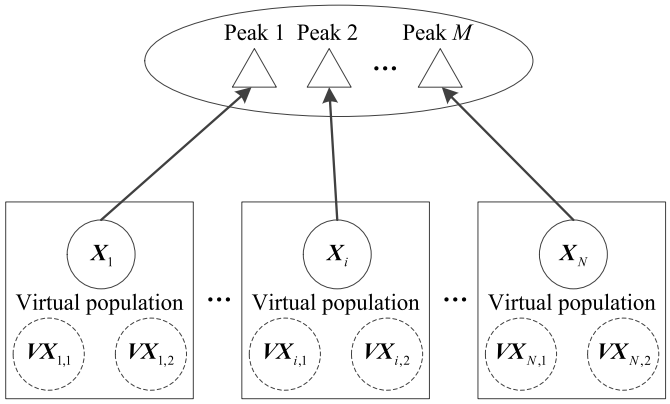
Digital Object Identifier 10.1109/TEVC.2019.2944180

I. Introduction

MULTIMODAL optimization problems (MMOPs) refer to problems that have multiple optimal solutions, which are common in real-world applications [1], [2]. Generally speaking, each optimum in MMOPs is also regarded as a peak and the algorithms are required to locate multiple peaks and to refine the solution accuracy on the found peaks.

Evolutionary algorithms (EAs) are a kind of promising approaches that have been widely used for solving optimization problems with a single global optimum [3]–[5]. However, when extending the conventional EAs to solve MMOPs, the multimodal EAs should maintain sufficient population diversity to find multiple peaks. To this aim, niching methods are popular approaches to extend EAs for MMOPs [6]. Niching methods preserve the diversity by dividing the population into several subpopulations. Those individuals in the same subpopulation are expected to lie in the region of the same peak. In this way, each subpopulation evolves independently to track a peak and finally the population can find multiple peaks. However, how to divide the population into subpopulations properly is a difficult issue. Most of the existing niching methods still have diversity challenge in locating as many peaks as possible and have accuracy challenge in refining the solution accuracy on the found peaks.

In order to further enhance EA's performance in solving MMOPs, we propose a novel distributed individuals for multiple peaks (DIMP) framework for extending EAs to solve MMOPs efficiently. The idea of DIMP is to let each individual act as a distributed unit to track a peak, which can avoid the difficulty of population division and also can well maintain sufficient diversity to locate more peaks. As differential evolution (DE) is a simple yet effective algorithm that has been widely studied in both single optimum optimization problems [7], [8] and MMOPs [9]–[16], we further propose a DIMP-based DE, termed as distributed individuals DE (DIDE), for more efficiently solving MMOPs in this article. However, as each individual acts as a distributed unit in the DIMP framework, the mutation operation in DIDE may be difficult to conduct because the mutation operation in DE often requires the guidance/difference from the other individuals in the same unit. To address this problem, we construct a virtual population for each individual to assist the mutation operation. The virtual population includes some virtual individuals that



N is the population size.
$\boldsymbol{X}_i$ is the i^{th} individual in the population.
$\boldsymbol{VX}_{i,1}$ and $\boldsymbol{VX}_{i,2}$ are the virtual individuals in the virtual population of $\boldsymbol{X}_i$.

Fig. 1. Basic framework of DIDE.

Algorithm 1 Original DE

1: Randomly initialize the population with N individuals;
2: **For** each individual $\boldsymbol{X}_i$
3: $\boldsymbol{V}_i = \boldsymbol{X}_{r1} + F \times (\boldsymbol{X}_{r2} - \boldsymbol{X}_{r3})$ where $r1, r2, r3 \in \{1, 2, \ldots, N\}, r1 \neq r2 \neq r3 \neq i$;
4: $rnbr_i = \text{Random}(1, D)$;
5: **For** each dimension d
6: $U_i^d = \begin{cases} V_i^d, & \text{if Random } (0, 1) \leq CR \text{ or } d == rnbr_i \\ X_i^d, & \text{otherwise} \end{cases}$;
7: **End For**
8: **If** $f(\boldsymbol{U}_i) \geq f(\boldsymbol{X}_i)$ /*For maximization problems*/
9: $\boldsymbol{X}_i = \boldsymbol{U}_i$;
10: **End If**
11: **End For**
12: Terminate if the termination criterion is met, otherwise jump back to Step 2;

are generated based on the individual. In this way, the mutation operation of each individual can be carried out by the guidance/difference of the virtual individuals in its corresponding virtual population. In order to generate a promising virtual population, the construction of virtual population is controlled by an adaptive range adjustment (ARA) strategy, where the range of the virtual individuals is initially large and gradually shrinks. Therefore, the diversity of the virtual population is high in the beginning to help the individual explore the search space sufficiently to locate a peak. With the range of virtual population gradually shrinks, the individual can gradually approach the peak. The basic framework of DIDE is shown in Fig. 1.

Based on the DIMP framework, the DIDE algorithm adopts a lifetime mechanism and an elite learning mechanism (ELM) to further cope with the diversity and accuracy challenges in solving MMOPs, respectively. Lifetime mechanism is inspired by the natural phenomenon that every organism will gradually age and has a limited lifespan. Biologists find that this phenomenon plays an important role in maintaining population diversity [17], [18]. In DIDE, each individual acts as a distributed unit and is also assigned a lifetime. During its lifetime, the individual can sufficiently explore the search space to locate a peak and then gradually approach it according to the DIMP framework and the ARA strategy. Once it exhausts its lifetime, the individual will be reinitialized and start a new lifetime. The reinitialized individuals can bring further diversity to help find more peaks. Note that for each life-exhausted individual, if it has a good enough fitness that meets an access criterion, it is regarded as an elite solution and will be added to an archive.

Different from the lifetime mechanism, the ELM aims to refine the accuracy of those elite solutions in the archive. Generally, the elite solutions already locate some peaks, which are promising candidates for the further elite learning to improve the accuracy. However, some elite solutions in the archive may locate the same peak, so that it is unnecessary to carry out elite learning on every elite solution. To avoid this problem, clustering method is conducted on all the elite solutions and only the elite solution with the best fitness in each cluster is selected as the representative to carry out elite learning. The ELM exploits around the selected elite solutions based on the Gaussian distribution. Moreover, an exponential descent strategy is proposed to adaptively adjust the standard deviation of the Gaussian distribution, aiming to make a sufficient exploitation for refining the solution accuracy.

In summary, the proposed DIDE algorithm is based on the DIMP framework that lets each individual act as a distributed unit to track a peak. The virtual population and its ARA strategy make the DIDE work efficiently under the DIMP framework. Moreover, the lifetime mechanism in DIDE helps to enhance the population diversity while the ELM in DIDE helps to refine the solution accuracy. The performance of DIDE is evaluated on the widely used CEC'2013 benchmark set that contains 20 multimodal test functions. Compared with the state-of-the-art multimodal optimization algorithms, DIDE obtains the overall better performance.

The rest of this article is organized as follows. Section II presents the DE and the multimodal optimization algorithms in the literature. Section III presents the DIDE algorithm and its novelties in detail. Section IV presents the experimental study. Finally, Section V presents the conclusion.

II. Differential Evolution and Multimodal Optimization Algorithms

A. Differential Evolution

The original DE algorithm is shown in Algorithm 1 [19]. First, the population is randomly initialized. Then for each individual $\boldsymbol{X}_i$ in every generation, the mutation, crossover, and selection operations are carried out sequentially. The mutation operation is shown in step 3, where $\boldsymbol{X}_{r1}$, $\boldsymbol{X}_{r2}$, and $\boldsymbol{X}_{r3}$ are three different individuals (also different from $\boldsymbol{X}_i$) randomly picked from the population. Moreover, F is a parameter named scaling factor. The crossover operation is shown in steps 4–7 that selects the information in each dimension according to a crossover rate CR. The $rnbr_i$ is a random integer for individual $\boldsymbol{X}_i$ and ranges in $\{1, 2, \ldots, D\}$ where D is the dimension size of the optimization problem. It aims to guarantee that at least one dimension is from the mutant offspring. The selection operation is shown in steps 8–10 that greedily selects the

one with better (e.g., higher for maximization problems) fitness between $\boldsymbol{X}_i$ and $\boldsymbol{U}_i$ for the next generation. The algorithm terminates if the termination criterion is met, otherwise it will jump back to step 2 and start a new generation.

B. Multimodal Optimization Algorithms

Existing strategies for enhancing EAs in solving MMOPs can be roughly classified into the following three categories, i.e., niching method, novel evolutionary operator, and multiobjective approach.

1) Niching Method: The principle of niching method is dividing the population into several diverse subpopulations to preserve the population diversity, with an aim to locate multiple peaks. In the literature, niching methods are widely researched, in which crowding [9] and speciation [10], [20] are two well-known frameworks. In crowding framework, the offspring replaces its most similar individual in a subpopulation. The subpopulation size is determined by a parameter named crowding factor. Speciation framework divides the population according to a species radius. The formation of subpopulations always requires some niche parameters (e.g., subpopulation size and species radius). However, the algorithm's performance is sensitive to these parameters.

To reduce the parameter sensitivity, clustering methods are incorporated into the algorithms for population division [11], [12], which employ a less sensitive parameter, i.e., the cluster size. Wang *et al.* [21] adopted the affinity propagation clustering, which did not require the cluster size to be predetermined. In addition, some topology-based niching methods are proposed for adaptive population division. They detect whether there is a valley between two individuals on the fitness landscape. If so, the two individuals are in the region of different peaks and should be divided into different subpopulations. Sampling methods are adopted in [13] and [22] to capture the fitness landscape, but they require extra function evaluations (FEs). Li and Tang [23] proposed a history-based topological speciation technique, where the search history was utilized to capture the fitness landscape without extra FEs.

2) Novel Evolutionary Operator: A number of researches proposed novel evolutionary operators for adapting EAs to MMOPs. Biswas *et al.* [14] designed a parent-centric normalized neighborhood mutation operator for DE, which restricted the randomness without inhibiting the explorative power. They also proposed locally informative DEs [15] that adopted two novel individual generation schemes to assist the mutation operation. Wang *et al.* [16] proposed a dual-strategy DE that adopted two mutation operators and each individual can adaptively select one of them to balance exploration and exploitation. Zhao *et al.* [24] proposed a novel mutation operator for DE that utilized both the local guidance of the subpopulation and the global guidance of the whole population. Qu *et al.* [25] proposed a locally informed particle swarm optimization that used several local best solutions rather than the global best solution to guide the search of each particle.

3) Multiobjective Approach: As multiobjective optimization [26]–[29] and many-objective optimization [30] have been widely studied, some researchers also proposed to transform MMOP into multiobjective optimization problem. In this sense, two optimization objectives are commonly considered. One is the fitness value and the other is a diversity indicator. For example, gradient information is utilized to construct the diversity indicator in [31] and [32] while distance metric is adopted in [33]. Moreover, Cheng *et al.* [34] adopted a grid-based diversity indicator. Instead of setting fitness value and diversity indicator as two optimization objectives, Wang *et al.* [35] designed two conflicting objectives in each dimension. In this way, multiple optimal solutions in an MMOP correspond to the Pareto optimal solutions of the transformed multiobjective optimization problem. The recent multiobjective approaches in [34] and [35] perform well on low-dimensional problems, but they are still not good enough on high-dimensional problems.

III. Distributed Individuals Differential Evolution

Our proposed DIDE algorithm, based on the DIMP framework, lets each individual conduct its mutation operation based on the virtual population to keep the distributed search of individuals, which can maintain sufficient population diversity to locate as many peaks as possible. In addition, the lifetime mechanism is proposed to further enhance the population diversity for locating more peaks. Lastly, the ELM is proposed to refine the solution accuracy on the found peaks.

A. Virtual Population-Based Mutation

Based on the DIMP framework, each individual in DIDE acts as a distributed unit, aiming to maintain sufficient population diversity. However, the mutation operation of an individual in DE often requires the information from the other individuals in the same unit, which brings difficulties for the distributed individuals to conduct the mutation operation. Therefore, in the proposed DIDE algorithm, each individual constructs a virtual population to assist the mutation operation. That is, the mutation operation for the individual $\boldsymbol{X}_i$ is designed as (1), where $\boldsymbol{VX}_{i,1}$ and $\boldsymbol{VX}_{i,2}$ are two different virtual individuals of $\boldsymbol{X}_i$

$$\boldsymbol{V}_i = \boldsymbol{X}_i + F \times \left(\boldsymbol{VX}_{i,1} - \boldsymbol{VX}_{i,2}\right). \tag{1}$$

The virtual individuals of $\boldsymbol{X}_i$ are randomly generated around $\boldsymbol{X}_i$ according to its virtual individual range $\boldsymbol{R}_i$, as

$$VX_i^d = \text{Random}\left(\max\left(X_i^d - \frac{R_i^d}{2}, L^d\right), \min\left(X_i^d + \frac{R_i^d}{2}, U^d\right)\right) \tag{2}$$

where $d \in \{1, 2, \ldots, D\}$ and D is the dimension size of the optimization problem. $\boldsymbol{L}$ and $\boldsymbol{U}$ are the lower bound and the upper bound of the search space, respectively. Generally, we adopt $X_i^d - R_i^d/2$ and $X_i^d + R_i^d/2$ as the lower bound and the upper bound of $\boldsymbol{X}_i$'s virtual individuals in each dimension d. However, in order to ensure that the virtual individuals do not exceed the search space, the lower bound and the upper bound are restricted to L^d and U^d, respectively if $X_i^d - R_i^d/2$ and $X_i^d + R_i^d/2$ exceed the search space. Each time the mutation is conducted, the virtual individuals of each individual

Algorithm 2 ARA Strategy

1: **For** each individual $\boldsymbol{X}_i$ in the population
2: **If** $cg_i \geq mcg$
3: $\boldsymbol{R}_i = \boldsymbol{R}_i/2$;
4: $cg_i = 0$;
5: $ht_i = ht_i + 1$;
6: **End If**
7: **End For**

Algorithm 3 Lifetime Mechanism

1: **For** each individual $\boldsymbol{X}_i$ in the population
2: **If** $ht_i \geq mht$
3: $rank = 1$;
4: **For** each other individual $\boldsymbol{X}_j$
5: **If** $f(\boldsymbol{X}_i) < f(\boldsymbol{X}_j)$ /*For maximization problems*/
6: $rank = rank + 1$;
7: **End If**
8: **End For**
9: **If** $rank \leq N \times at$ /*N is the population size*/
10: Add $\boldsymbol{X}_i$ to the archive;
11: **End If**
12: Re-initialize $\boldsymbol{X}_i$;
13: $\boldsymbol{R}_i = \boldsymbol{U} - \boldsymbol{L}$; $cg_i = 0$; $ht_i = 0$;
14: **End If**
15: **End For**

are regenerated according to (2). Note that the virtual individuals are only used to assist the mutation operation and only their position information is used in (1). Therefore, the virtual individuals do not need to be evaluated, with no extra FEs needed.

The setting of $\boldsymbol{R}$ is important in DIDE. A large $\boldsymbol{R}$ enhances the exploration but brings difficulty for the exploitation, while a small $\boldsymbol{R}$ reverses. In order to balance the exploration and the exploitation, $\boldsymbol{R}$ is associated with each individual and is adaptively adjusted by the ARA strategy in every generation. A large $\boldsymbol{R}$ is initialized at the beginning to help the individual explore the search space sufficiently to locate a peak. During the evolutionary process, the $\boldsymbol{R}$ gradually decreases to help exploit the peak. Therefore, the individual can gradually approach the peak.

In detail, each individual's $\boldsymbol{R}$ is initialized to ($\boldsymbol{U}$–$\boldsymbol{L}$), which is the entire search range. Then, during the evolutionary process, the $\boldsymbol{R}$ is adaptively adjusted by the ARA strategy (shown in Algorithm 2) to gradually decrease. Herein, the adaptive adjustment of $\boldsymbol{R}$ is according to whether the $\boldsymbol{R}$ can continuously help to improve the individual. We use an indicator *cg* to record the consecutive generations that the individual fails to be improved (i.e., is not replaced by the offspring in the selection operation). If the individual's *cg* reaches *mcg* (a parameter named *maximal consecutive generations*), then the ARA strategy halves the $\boldsymbol{R}$ to enhance the exploitation and *cg* is reset to 0, as steps 3 and 4 in Algorithm 2. Meanwhile, the halved times *ht* will increase by one (step 5), where *ht* records the number of times that the $\boldsymbol{R}$ has been halved. Herein, each individual's *ht* is initialized to 0 in each lifetime and is used to control the lifetime mechanism (details refer to Section III-B). Notice that each individual $\boldsymbol{X}_i$ has its independent $\boldsymbol{R}$, *cg*, and *ht* (i.e., $\boldsymbol{R}_i$, cg_i, and ht_i as shown in Algorithm 2). However, to make the description clearer, we simply use $\boldsymbol{R}$, *cg*, and *ht* in the above context.

The larger the search space, the more generations are needed for an individual to make a sufficient search. Therefore, *mcg* is set adaptively based on the dimension size of the optimization problem, as

$$mcg = 10 \times 2^{\lfloor D/10 \rfloor + 1}. \tag{3}$$

B. Lifetime Mechanism

According to the DIMP framework and the ARA strategy, each individual can locate a peak and gradually approach it. However, the initial population may be unable to locate all peaks. For example, even though each individual can locate a peak, the population still cannot locate all peaks if the population size is less than the number of peaks. For another example, the fitness landscapes of different peaks may be different. In this case, the individuals are more likely to move to those peaks with large region and gentle slope, while those peaks with small region and steep slope are difficult to explore.

Therefore, inspired by the aging phenomenon in nature, we propose a lifetime mechanism shown in Algorithm 3 to let DIDE have opportunities to locate more peaks. In order to activate the lifetime mechanism, a maximum halved times *mht* is defined. If an individual's *ht* reaches *mht*, the lifetime of this individual is exhausted. If the life-exhausted individual meets an access criterion (details refer to the next paragraph and steps 3–11 in Algorithm 3), it is regarded as an elite solution and will be added to the archive. After the judgement, the individual will be reinitialized, together with its $\boldsymbol{R}$, *cg*, and *ht*, as steps 12 and 13 in Algorithm 3. In this way, the archive will keep the individual's elite knowledge in each lifetime, while the individual's new lifetime will bring new information to the population that will greatly enhance the diversity to locate more peaks.

Herein, the access criterion for judging a life-exhausted individual and adding it to the archive (as steps 3–11 in Algorithm 3) is described and discussed. For a life-exhausted individual, it may locate a global peak or a local peak. However, it is only useful to add those life-exhausted individuals around the global peaks to the archive and refine their accuracies via ELM, where ELM is detailed described in Section III-C. Therefore, a simple access criterion is proposed to filter out those life-exhausted individuals around the local peaks. Before adding a life-exhausted individual to the archive, we first check the rank of its fitness in the entire population. If it ranks top *at* (a parameter named *access threshold* ranging in (0, 1]) of the population, we regard it as an elite solution and add it to the archive. Otherwise, it will be filtered. The parameter analysis of *at* (details refer to Section IV-E) shows that setting *at* to 0.8 can obtain a good tradeoff between filtering the life-exhausted individuals around local peaks and preserving those around global peaks, helping DIDE to have a good performance. Therefore, *at* is set to 0.8 in DIDE.

C. Elite Learning Mechanism

The elite solutions are stored in the archive. Each elite solution already locates a peak but it may still not meet the

accuracy requirement. Therefore, we propose the ELM shown in Algorithm 4 to refine the accuracy of the elite solutions. It should be noticed that some elite solutions in the archive may actually locate the same peak. In this case, it is unnecessary to carry out elite learning on every elite solution. In order to avoid repeating similar work, clustering method can be a promising solver because it can cluster those solutions locating the same peak. Then we can select one representative solution from each cluster to carry out elite learning. Among the well-known clustering methods, the k-means clustering requires the number of clusters to be predetermined but the number of peaks in MMOPs is not known in advance. Therefore, the nonparametric mean-shift clustering [36], [37] is employed in DIDE. The principle of mean-shift clustering is iteratively shifting each point to a higher density position until convergence. Those points whose shifted points converge to the same position belong to the same cluster. The widely used Gaussian kernel function is adopted to determine the weight of points for estimating the density. The procedure of the mean-shift clustering is shown in Algorithm S.1 in the supplementary material. The Gaussian kernel function requires a parameter named bandwidth. According to the experimental tests, we find that the setting of bandwidth is not sensitive that the value ranging in [0.001, 0.01] does not have a significant influence on DIDE's performance. Therefore, bandwidth is simply set to 0.001 in DIDE. One note is that the clustering operation only need to be carried out in the generation where there is/are new elite solution(s) added to the archive by the lifetime mechanism.

After the clustering, each cluster may correspond to a peak. In each cluster, only the solution with the best fitness is selected for elite learning. Each selected solution actually refers to a solution in the archive (i.e., $\boldsymbol{A}_i$). The elite learning of $\boldsymbol{A}_i$ is according to (4), which is based on the Gaussian distribution with $\boldsymbol{A}_i$ as the mean and a standard deviation σ_i. In order to enhance the efficiency, two elite learning solutions are generated each time. The better one of the two elite learning solutions will replace $\boldsymbol{A}_i$ if it has better fitness than $\boldsymbol{A}_i$

$$\boldsymbol{ELS}_i = \text{Gaussian}(\boldsymbol{A}_i, \sigma_i). \tag{4}$$

A fixed σ_i cannot perform well in various cases. Therefore, an exponential descent strategy (steps 11–19 in Algorithm 4) is proposed to adaptively adjust σ_i. The σ_i is initialized to σ_{ini}. A stagnation counter sc_i is used to record the consecutive times that the elite learning solution cannot replace $\boldsymbol{A}_i$. If sc_i reaches a descent threshold dt, divide σ_i by 10 and reset sc_i to 0. The elite learning process of $\boldsymbol{A}_i$ will stop until σ_i is smaller than σ_{termin}. The exponential descent strategy helps to refine the solution accuracy sufficiently.

In addition, MMOPs may have some peaks that are very difficult to exploit. In this case, only one elite learning process for $\boldsymbol{A}_i$ may be not sufficient to obtain high accuracy solution. To address this problem, we can restart a new elite learning process for $\boldsymbol{A}_i$. That is, as steps 20–24 in Algorithm 4, when σ_i is smaller than σ_{termin}, $\boldsymbol{A}_i$ finishes its current elite learning process. However, if it is worse than the best solution in the archive (denoted as $\boldsymbol{A}_{best}$), we think that $\boldsymbol{A}_i$ has not sufficiently exploited the peak yet and it will restart a new elite learning process. In detail, for a maximization problem, if $f(\boldsymbol{A}_{best}) > f(\boldsymbol{A}_i)$, then σ_i will be reset to σ_{ini} for restarting a new elite learning process.

Algorithm 4 ELM

1: **If** there is/are new elite solution(s) added to the archive
2: Initialize its/their σ and sc to σ_{ini} and 0, respectively;
3: Cluster the solutions in the archive by mean-shift clustering;
4: **End If**
5: **For** each cluster
6: Its best solution refers to a solution $\boldsymbol{A}_i$ in the archive;
7: **If** $\sigma_i \geq \sigma_{termin}$ /*$\boldsymbol{A}_i$ carries out the elite learning process*/
8: Generate two elite learning solutions $\boldsymbol{ELS}_{i,1}$ and $\boldsymbol{ELS}_{i,2}$ according to (4) and evaluate their fitness values; /*For maximization problems*/
9: Replace $\boldsymbol{A}_i$ with $\boldsymbol{ELS}_{i,1}$ if $f(\boldsymbol{ELS}_{i,1}) > f(\boldsymbol{A}_i)$;
10: Replace $\boldsymbol{A}_i$ with $\boldsymbol{ELS}_{i,2}$ if $f(\boldsymbol{ELS}_{i,2}) > f(\boldsymbol{A}_i)$;
11: **If** replacement does not happen
12: $sc_i = sc_i + 2$;
13: **If** $sc_i \geq dt$
14: $\sigma_i = \sigma_i/10$;
15: $sc_i = 0$;
16: **End If**
17: **Else**
18: $sc_i = 0$; /*reset sc because the replacement happens*/
19: **End If**
20: **Else** /*$\sigma_i < \sigma_{termin}$, i.e., $\boldsymbol{A}_i$ finishes the elite learning process*/
21: **If** $f(\boldsymbol{A}_{best}) > f(\boldsymbol{A}_i)$ /*For maximization problems*/
22: $\sigma_i = \sigma_{ini}$; /*restart a new elite learning process for $\boldsymbol{A}_i$*/
23: Go to Step 8;
24: **End If**
25: **End If**
26: **End For**

Algorithm 5 DIDE

Randomly initialize each individual in the population;
For each individual $\boldsymbol{X}_i$
  $\boldsymbol{R}_i = \boldsymbol{U} - \boldsymbol{L}$; $cg_i = 0$; $ht_i = 0$;
End For
For each individual $\boldsymbol{X}_i$
  Generate two virtual individuals $\boldsymbol{VX}_{i,1}$ and $\boldsymbol{VX}_{i,2}$ according to (2);
  $\boldsymbol{V}_i = \boldsymbol{X}_i + F \times (\boldsymbol{VX}_{i,1} - \boldsymbol{VX}_{i,2})$;
  $rnbr_i = \text{Random}(1, D)$;
  For each dimension d
    $U_i^d = \begin{cases} V_i^d, & \text{if Random } (0, 1) \leq CR \text{ or } d == rnbr_i \\ X_i^d, & \text{otherwise} \end{cases}$;
  End For
  If $f(\boldsymbol{U}_i) \geq f(\boldsymbol{X}_i)$ /*For maximization problems*/
    $\boldsymbol{X}_i = \boldsymbol{U}_i$;
    $cg_i = 0$;
  Else
    $cg_i = cg_i + 1$;
  End If
End For
Carry out the ARA strategy according to Algorithm 2;
Carry out the lifetime mechanism according to Algorithm 3;
Carry out the ELM according to Algorithm 4;
Terminate if the termination criterion is met, otherwise jump back to Step 5;

D. Complete Procedure of Distributed Individuals Differential Evolution

The complete procedure of DIDE is shown in Algorithm 5. First, each individual is randomly initialized. Then each

individual's $\boldsymbol{R}$, cg, and ht are also initialized. In each generation, each individual $\boldsymbol{X}_i$ generates two virtual individuals according to (2) (step 6) and carries out the mutation operation according to (1) (step 7). If the position of $\boldsymbol{V}_i$ is out of the search space, it will be restricted to the corresponding boundary. The crossover operation (steps 8–11) is the same as the original DE in Algorithm 1. After that, an offspring $\boldsymbol{U}_i$ is generated. The selection operation (steps 12–17) is carried out by greedily selecting the one with better fitness between $\boldsymbol{U}_i$ and $\boldsymbol{X}_i$ for the next generation. Meanwhile, the cg of $\boldsymbol{X}_i$ is updated to assist the ARA strategy. After all individuals finishing a generation, the ARA strategy, lifetime mechanism, and ELM are carried out sequentially (steps 19–21). DIDE terminates if the termination criterion is met, otherwise it will jump back to step 5 and start a new generation.

IV. Experimental Study

A. Benchmark Functions and Performance Metrics

The widely used CEC'2013 benchmark set [38] is adopted to evaluate the performance of DIDE in solving MMOPs. The CEC'2013 benchmark set contains 20 multimodal test functions and the basic properties of these 20 functions are shown in Table S.I. in the supplementary material. All of the functions are maximization problems. F_1, F_2, and F_3 are simple 1-D functions. F_4 and F_5 are 2-D functions that are not scalable. $F_6 - F_{10}$ are scalable functions of 2-D or 3-D. $F_{11} - F_{20}$ are complex composition functions constructed by several basic functions with different properties. Among these 20 functions, $F_1 - F_{17}$ are low-dimensional functions of 1-D–5-D while $F_{18} - F_{20}$ are high-dimensional functions of 10-D or 20-D. More details of the CEC'2013 benchmark set can be referred to [38].

Two metrics are adopted for the performance evaluation, which are peak ratio (PR) and success rate (SR). Given a maximum number of FEs (MaxFEs) and an accuracy level ε, PR represents the average percentage of global peaks that found over multiple runs, formulated as

$$\mathrm{PR} = \frac{\sum_{i=1}^{\mathrm{NR}} \mathrm{NPF}_i}{\mathrm{TNP} \times \mathrm{NR}} \tag{5}$$

where NR is the number of runs, NPF_i is the number of global peaks found in the ith run, and TNP is the total number of global peaks in the optimization problem.

SR represents the percentage of the successful runs among multiple runs, formulated as

$$\mathrm{SR} = \frac{\mathrm{NSR}}{\mathrm{NR}} \tag{6}$$

where NSR is the number of successful runs among total NR runs. A successful run is defined as a run where all global peaks are found. More details of these two performance metrics can be referred to [38].

Three accuracy levels, ε = 1.0E–03, ε = 1.0E–04, and ε = 1.0E–05, are adopted in the experiments. One note is that we mainly discuss the experimental results at $\varepsilon = 1.0\text{E–}04$, as in [16] and [35], while the experimental results at $\varepsilon = 1.0\text{E–}03$ and $\varepsilon = 1.0\text{E–}05$ are presented in the supplementary material. The settings of MaxFEs on different functions are shown in Table I, the same as the suggestion in [38].

TABLE I
Setting of MaxFEs

Function	MaxFEs
F_1–F_5	5.0E+04
F_6–F_7	2.0E+05
F_8–F_9	4.0E+05
F_{10}–F_{13}	2.0E+05
F_{14}–F_{20}	4.0E+05

B. Compared Algorithms and Parameter Configurations

In this article, our proposed DIDE algorithm is compared with 13 state-of-the-art multimodal optimization algorithms, which are CDE [9], SDE [10], NCDE, NSDE [11], MOMMOP [35], LIPS [25], NMMSO [39], LoICDE, LoISDE [15], PNPCDE [14], EMO-MMO [34], self-CCDE, and self-CSDE [12]. The population size is set to 100 for all compared algorithms and our proposed DIDE algorithm, except for MOMMOP, NMMSO, and EMO-MMO because they have their special setting schemes of population size to deal with the CEC'2013 benchmark set. The configurations of the other parameters in the compared algorithms are all based on their corresponding references.

In DIDE, the scaling factor F and the crossover rate CR are set to 0.3 and 0.9, respectively. In lifetime mechanism, the maximum halved times mht is 10. In ELM, the descent threshold dt is 40 while the σ_{ini} and σ_{termin} are 1.0E–04 and 1.0E–10, respectively.

For fair comparison, 50 independent runs are conducted and the average results are reported. The Wilcoxon rank-sum test at the widely used significance level $\alpha = 0.05$ is adopted to statistically evaluate the PR results in the 50 runs. If DIDE is significantly better than a compared algorithm, the result is "+." If DIDE is significantly worse, the result is "−." If the performances of DIDE and the compared algorithm are not significantly different, the result is "≈."

C. Comparison With State-of-the-Art Algorithms

The experimental results of PR and SR on the CEC'2013 benchmark set at the accuracy level $\varepsilon = 1.0\text{E–}04$ are shown in Table II. Those PR results that are the best among DIDE and the compared algorithms are bolded. Form the table, DIDE's performance is the best on 16 functions out of all the 20 functions, which is the most among all tested algorithms.

On F_1–F_6 and F_{10}, DIDE can stably find all the peaks (i.e., the SR results are 1.000). Among all the compared algorithms, only EMO-MMO can achieve the same performance as DIDE. MOMMOP fails on F_4 while NMMSO fails on F_6. The other algorithms fail on at least two functions. On F_7–F_9, MOMMOP, NMMSO, and EMO-MMO obtain very good performance, being better than DIDE. However, the performance of DIDE on F_7–F_9 is still very promising and is significantly better than the other ten compared algorithms.

TABLE II
EXPERIMENTAL RESULTS OF PR AND SR ON THE CEC'2013 BENCHMARK SET AT THE ACCURACY LEVEL $\varepsilon = 1.0\text{E}{-}04$

Func	DIDE		CDE		SDE		NCDE		NSDE		MOMMOP		LIPS	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F_1	**1.000**	1.000	**1.000**(≈)	1.000	0.700(+)	0.420	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	0.960(+)	0.920
F_2	**1.000**	1.000	**1.000**(≈)	1.000	0.840(+)	0.680	**1.000**(≈)	1.000	0.640(+)	0.480	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F_3	**1.000**	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F_4	**1.000**	1.000	0.985(≈)	0.940	0.325(+)	0.000	**1.000**(≈)	1.000	0.250(+)	0.000	0.985(≈)	0.940	**1.000**(≈)	1.000
F_5	**1.000**	1.000	**1.000**(≈)	1.000	0.970(≈)	0.940	**1.000**(≈)	1.000	0.840(+)	0.680	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F_6	**1.000**	1.000	0.640(+)	0.020	0.058(+)	0.000	0.327(+)	0.000	0.051(+)	0.000	**1.000**(≈)	1.000	0.204(+)	0.000
F_7	0.921	0.040	0.719(+)	0.000	0.034(+)	0.000	0.558(+)	0.000	0.028(+)	0.000	**1.000**(−)	1.000	0.216(+)	0.000
F_8	0.692	0.000	0.530(+)	0.000	0.012(+)	0.000	0.563(+)	0.000	0.007(+)	0.000	**1.000**(−)	1.000	0.048(+)	0.000
F_9	0.571	0.000	0.269(+)	0.000	0.005(+)	0.000	0.233(+)	0.000	0.005(+)	0.000	**1.000**(−)	0.920	0.044(+)	0.000
F_{10}	**1.000**	1.000	**1.000**(≈)	1.000	0.140(+)	0.000	0.990(+)	0.880	0.097(+)	0.000	**1.000**(≈)	1.000	0.758(+)	0.000
F_{11}	**1.000**	1.000	0.667(+)	0.000	0.217(+)	0.000	0.670(+)	0.000	0.170(+)	0.000	0.710(+)	0.040	0.840(+)	0.320
F_{12}	**1.000**	1.000	0.000(+)	0.000	0.140(+)	0.000	0.105(+)	0.000	0.123(+)	0.000	0.955(+)	0.660	0.493(+)	0.000
F_{13}	0.987	0.920	0.667(+)	0.000	0.233(+)	0.000	0.630(+)	0.000	0.170(+)	0.000	0.667(+)	0.000	0.657(+)	0.040
F_{14}	**0.773**	0.020	0.667(+)	0.000	0.180(+)	0.000	0.640(+)	0.000	0.157(+)	0.000	0.667(+)	0.000	0.567(+)	0.000
F_{15}	**0.748**	0.000	0.528(+)	0.000	0.103(+)	0.000	0.283(+)	0.000	0.113(+)	0.000	0.618(+)	0.000	0.258(+)	0.000
F_{16}	**0.667**	0.000	**0.667**(≈)	0.000	0.107(+)	0.000	0.413(+)	0.000	0.077(+)	0.000	0.630(+)	0.000	0.233(+)	0.000
F_{17}	**0.593**	0.000	0.008(+)	0.000	0.085(+)	0.000	0.138(+)	0.000	0.040(+)	0.000	0.505(+)	0.000	0.138(+)	0.000
F_{18}	**0.667**	0.000	0.177(+)	0.000	0.013(+)	0.000	0.177(+)	0.000	0.003(+)	0.000	0.497(+)	0.000	0.050(+)	0.000
F_{19}	**0.543**	0.000	0.000(+)	0.000	0.063(+)	0.000	0.028(+)	0.000	0.000(+)	0.000	0.230(+)	0.000	0.000(+)	0.000
F_{20}	**0.355**	0.000	0.000(+)	0.000	0.000(+)	0.000	0.155(+)	0.000	0.000(+)	0.000	0.125(+)	0.000	0.000(+)	0.000
+ (DIDE is significantly better)			13		18		15		18		10		16	
≈ (Not significantly different)			7		2		5		2		7		4	
− (DIDE is significantly worse)			0		0		0		0		3		0	
Func	NMMSO		LoICDE		LoISDE		PNPCDE		EMO-MMO		Self_CCDE		Self_CSDE	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F_1	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F_2	**1.000**(≈)	1.000	**1.000**(≈)	1.000	0.240(+)	0.040	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F_3	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F_4	**1.000**(≈)	1.000	0.950(+)	0.800	0.250(+)	0.000	0.185(+)	0.000	**1.000**(≈)	1.000	0.995(≈)	0.980	0.675(+)	0.200
F_5	**1.000**(≈)	1.000	**1.000**(≈)	1.000	0.790(+)	0.580	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	0.960(+)	0.920
F_6	0.993(+)	0.880	0.981(+)	0.840	0.056(+)	0.000	0.942(+)	0.680	**1.000**(≈)	1.000	0.934(+)	0.340	0.686(+)	0.120
F_7	**1.000**(−)	1.000	0.701(+)	0.000	0.028(+)	0.000	0.697(+)	0.000	**1.000**(−)	1.000	0.672(+)	0.000	0.452(+)	0.000
F_8	0.904(−)	0.000	0.408(+)	0.000	0.012(+)	0.000	0.414(+)	0.000	**1.000**(−)	1.000	0.682(≈)	0.000	0.327(+)	0.000
F_9	0.975(−)	0.220	0.276(+)	0.000	0.005(+)	0.000	0.267(+)	0.000	0.950(−)	0.000	0.257(+)	0.000	0.106(+)	0.000
F_{10}	**1.000**(≈)	1.000	**1.000**(≈)	1.000	0.083(+)	0.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	0.992(+)	0.920
F_{11}	**1.000**(≈)	1.000	0.647(+)	0.000	0.167(+)	0.000	0.623(+)	0.000	**1.000**(≈)	1.000	0.780(+)	0.120	0.470(+)	0.000
F_{12}	0.983(+)	0.860	0.020(+)	0.000	0.125(+)	0.000	0.003(+)	0.000	**1.000**(≈)	1.000	0.263(+)	0.000	0.265(+)	0.000
F_{13}	0.990(≈)	0.940	0.440(+)	0.000	0.167(+)	0.000	0.430(+)	0.000	**0.997**(≈)	0.980	0.617(+)	0.000	0.387(+)	0.000
F_{14}	0.700(+)	0.000	0.603(+)	0.000	0.167(+)	0.000	0.317(+)	0.000	0.733(+)	0.060	0.640(+)	0.000	0.303(+)	0.000
F_{15}	0.620(+)	0.000	0.235(+)	0.000	0.125(+)	0.000	0.093(+)	0.000	0.595(+)	0.000	0.318(+)	0.000	0.145(+)	0.000
F_{16}	0.663(≈)	0.000	0.420(+)	0.000	0.167(+)	0.000	0.133(+)	0.000	0.657(≈)	0.000	0.603(+)	0.000	0.043(+)	0.000
F_{17}	0.478(+)	0.000	0.233(+)	0.000	0.055(+)	0.000	0.003(+)	0.000	0.335(+)	0.000	0.245(+)	0.000	0.038(+)	0.000
F_{18}	0.660(≈)	0.000	0.170(+)	0.000	0.157(+)	0.000	0.160(+)	0.000	0.327(+)	0.000	0.290(+)	0.000	0.000(+)	0.000
F_{19}	0.350(+)	0.000	0.003(+)	0.000	0.013(+)	0.000	0.000(+)	0.000	0.135(+)	0.000	0.080(+)	0.000	0.000(+)	0.000
F_{20}	0.170(+)	0.000	0.125(+)	0.000	0.060(+)	0.000	0.008(+)	0.000	0.080(+)	0.000	0.070(+)	0.000	0.000(+)	0.000
+	7		15		18		15		6		13		17	
≈	10		5		2		5		11		7		3	
−	3		0		0		0		3		0		0	

F_{11}–F_{20} are complex composition functions, where DIDE has great advantage compared with the other algorithms. On F_{11} and F_{12}, DIDE can stably find all the peaks. On F_{13}, although EMO-MMO's performance is the best, the performances of DIDE and EMO-MMO are not significantly different according to the Wilcoxon rank-sum test. On F_{14}–F_{20}, DIDE's performance is the best among all tested algorithms. EMO-MMO shows strong competitiveness on F_{11}–F_{13}, but it cannot deal with F_{14}–F_{20} very well. In detail, DIDE outperforms EMO-MMO on six functions (i.e., F_{14}, F_{15}, and F_{17}–F_{20}).

Particularly, DIDE's performance on F_{19} and F_{20} (two high-dimensional functions of 10-D and 20-D, respectively) is obviously better than the other algorithms. These two functions are more challenging than the other composition functions due to their high-dimension sizes. Many algorithms can only locate very few peaks or even cannot find any peak on both F_{19} and

TABLE III
EXPERIMENTAL RESULTS OF PR IN THE COMPONENT ANALYSIS OF LIFETIME MECHANISM AND ELM

Accuracy Level	F_1			F_2			F_3			F_4		
	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*
1.0E−03	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)
1.0E−04	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)
1.0E−05	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	**1.000**(≈)
Accuracy Level	F_5			F_6			F_7			F_8		
	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*
1.0E−03	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	**1.000**(≈)	0.974(+)	**0.922**	0.921(≈)	0.662(+)	**0.693**	0.016(+)	0.304(+)
1.0E−04	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	0.967(+)	0.974(+)	**0.921**	0.918(≈)	0.662(+)	**0.692**	0.001(+)	0.304(+)
1.0E−05	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	0.363(+)	0.974(+)	**0.920**	0.899(+)	0.662(+)	**0.689**	0.000(+)	0.304(+)
Accuracy Level	F_9			F_{10}			F_{11}			F_{12}		
	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*
1.0E−03	0.580	**0.643**(−)	0.238(+)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	0.667(+)	**1.000**(≈)	**1.000**	0.740(+)	0.960(+)
1.0E−04	0.571	**0.614**(−)	0.238(+)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	0.667(+)	**1.000**(≈)	**1.000**	0.580(+)	0.960(+)
1.0E−05	**0.561**	0.552(≈)	0.237(+)	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**	0.667(+)	**1.000**(≈)	**1.000**	0.363(+)	0.960(+)
Accuracy Level	F_{13}			F_{14}			F_{15}			F_{16}		
	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*
1.0E−03	**0.987**	0.667(+)	0.820(+)	**0.777**	0.667(+)	0.710(+)	**0.748**	0.573(+)	0.620(+)	**0.667**	**0.667**(≈)	**0.667**(≈)
1.0E−04	**0.987**	0.667(+)	0.820(+)	**0.773**	0.667(+)	0.710(+)	**0.748**	0.450(+)	0.620(+)	**0.667**	0.543(+)	**0.667**(≈)
1.0E−05	**0.957**	0.660(+)	0.820(+)	**0.733**	0.423(+)	0.710(≈)	**0.748**	0.160(+)	0.620(+)	**0.667**	0.003(+)	**0.667**(≈)
Accuracy Level	F_{17}			F_{18}			F_{19}			F_{20}		
	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*	DIDE	NE*	NLE*
1.0E−03	**0.593**	0.398(+)	0.465(+)	**0.667**	0.493(+)	0.643(+)	**0.543**	0.118(+)	0.463(+)	**0.355**	0.000(+)	0.340(≈)
1.0E−04	**0.593**	0.140(+)	0.465(+)	**0.667**	0.000(+)	0.643(+)	**0.543**	0.000(+)	0.463(+)	**0.355**	0.000(+)	0.305(+)
1.0E−05	**0.588**	0.005(+)	0.465(+)	**0.667**	0.000(+)	0.643(+)	**0.535**	0.000(+)	0.463(+)	**0.345**	0.000(+)	0.168(+)

NE* represents the DIDE without ELM; NLE* represents the DIDE without both lifetime mechanism and ELM.

F_{20}, i.e., CDE, SDE, NSDE, LIPS, LoISDE, PNPCDE, Self-CCDE, and Self-CSDE. Moreover, NCDE and LoICDE can only locate very few peaks on F_{19} while EMO-MMO can only locate very few peaks on F_{20}. DIDE' PR results on F_{19} and F_{20} are 0.543 and 0.355, respectively, while the best PR results among all the compared algorithms on these two functions are only 0.350 and 0.170, respectively.

In the bottom of each compared algorithm column, the numbers of functions that DIDE performs significantly better, not significantly different, and significantly worse, are counted and presented according to the results of Wilcoxon rank-sum test. Compared with any of the algorithms, the number of functions that DIDE is significantly better is more than the number of functions that DIDE is significantly worse, which shows the overall better performance of DIDE.

The experimental results of PR and SR on the CEC'2013 benchmark set at the accuracy level $\varepsilon = 1.0\text{E}{-}03$ and $\varepsilon = 1.0\text{E}{-}05$ are shown in Tables S.II and S.III in the supplementary material, respectively. They further verify the good performance of DIDE.

D. Component Analysis

The effect of DIDE's two components, lifetime mechanism and ELM, is discussed as follows. As described in Section III-B, one of the conditions for an individual to be added to the archive is that the individual exhausts its lifetime. If the lifetime mechanism is unemployed, no solution will be added to the archive and as a result, ELM will also not function. Therefore, we test the DIDE without ELM and the DIDE without both lifetime mechanism and ELM.

The experimental results are shown in Table III. The best results on each function are bolded. In order to save space, the DIDE without ELM is abbreviated as NE* (*N*o *E*LM) and the DIDE without both lifetime mechanism and ELM is abbreviated as NLE* (*N*o *L*ifetime nor *E*LM). Compared with NE*, DIDE has significantly higher PR on $F_{11}-F_{20}$, which shows that ELM can greatly refine the solution accuracy. However, NE* outperforms DIDE on F_9 at the accuracy level of 1.0E–03 and 1.0E–04. That is because the solutions found by those life-exhausted individuals have already met the accuracy requirement so that ELM cannot bring further improvement and may waste the FEs. NLE*'s performance on F_{11} and F_{16} is quite competitive. According to the ARA strategy in Algorithm 2, the decrease of $\boldsymbol{R}$ will be unlimited if the lifetime mechanism is unemployed, which encourages individuals' exploitation. As a result, the solution accuracy will be improved gradually. However, without the lifetime mechanism, the NLE* does not have sufficient diversity to locate more peaks. Therefore, DIDE still outperforms NLE* on many functions, particularly on F_7–F_9, with the cooperation of lifetime mechanism and ELM.

The experimental results on F_6–F_9 can further illustrate the effect of these two components. As shown in Fig. 2(a), the peaks in F_6 are "sharp," which are difficult to refine solution accuracy. Therefore, NE* performs poorly without ELM. As we can see from the table, NE*'s PR result on F_6 at the accuracy level of 1.0E–5 is only 0.363 while DIDE can find all the

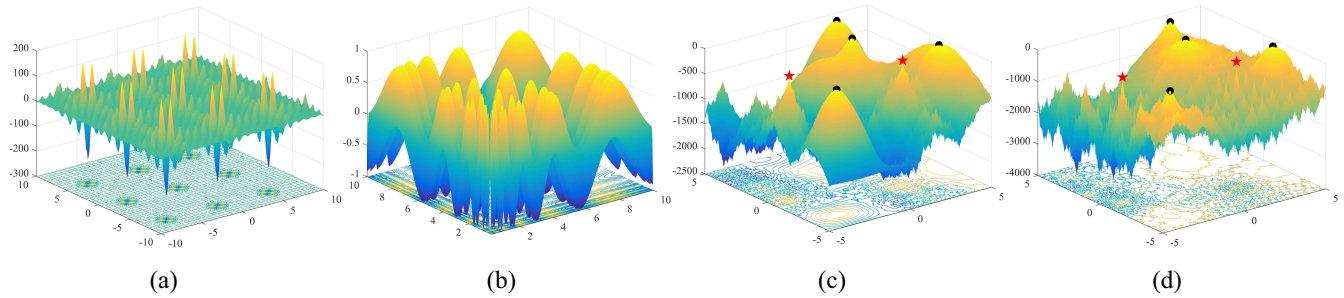


Fig. 2. Fitness landscapes of (a) F_6, (b) F_7, (c) F_{11}, and (d) F_{13}. In (c) and (d), the red stars and black circles mark the global peaks with different characteristics.

peaks (i.e., PR = 1.000). The experimental results on F_8 (the 3-D version of F_6) are more obvious, where NE* finds very few peaks. The peaks in F_7, shown in Fig. 2(b), distribute unevenly and the region of each peak is of different sizes. The randomly initialized individuals are more likely to move to those peaks that have large region. In this case, the lifetime mechanism brings the reinitialized individuals with new opportunities to explore those small-region peaks. As we can see in the table, NE* outperforms NLE* on F_7 and F_9 (the 3-D version of F_7).

In conclusion, the lifetime mechanism enhances the population diversity to locate more peaks while the ELM refines the solution accuracy on the found peaks. The lifetime mechanism and the ELM cooperatively balance exploration and exploitation, helping DIDE obtain overall good performance.

Moreover, the proposed DIMP framework is compared with two existing frameworks in solving MMOPs, i.e., crowding and speciation. In detail, we replace the DIMP framework in our proposed algorithm with the crowding and speciation frameworks, and test their performance. The experimental results of PR under these three frameworks are shown in Table S.IV in the supplementary material. We can see that the DIMP framework has better performance than the crowding and speciation frameworks.

E. Parameter Analysis

We divide the parameters in DIDE into two groups. The first group contains three basic DE parameters, i.e., the population size N, the scaling factor F, and the crossover rate CR. The second group contains five parameters, i.e., the maximum halved times mht and access threshold at in lifetime mechanism, and the descent threshold dt, σ_{ini}, and σ_{termin} in ELM.

For each parameter, five levels of values are tested. Since it is too time-consuming to test all possible combinations of parameter setting, design of experiments (DOE) is carried out to test the parameter setting in this article. Herein, we use Taguchi's method [40], [41] to conduct DOE based on orthogonal arrays. The parameter settings based on orthogonal arrays for the two groups of parameters are shown in Tables S.V and S.VI in the supplementary material, respectively. Both of them contain 25 tests. When testing a group of parameters, the other parameters are set as adopted in DIDE. The experimental results of PR for the two groups of parameters at the accuracy level ε = 1.0E–03, 1.0E–04, and 1.0E–05 are shown in Tables S.VII–S.XII in the supplementary material, respectively. Note that the adopted setting in DIDE is not included in Tables S.V and S.VI in the supplementary material, and its experimental results are shown in the column named "adopted" in Tables S.VII-S.XII in the supplementary material. From these tables, we can see that compared with the other settings, the number of functions that the DIDE with adopted setting performs significantly better is more than the number of functions that it performs significantly worse, showing that the adopted parameter setting is promising.

Moreover, the Friedman test with Bonferroni–Dunn procedure [42] is adopted to test the robustness of DIDE. Friedman test can rank the performance of each parameter setting on each benchmark function, and Bonferroni–Dunn procedure can further detect whether the performances of the parameter settings in solving all the 20 benchmark functions are significantly different. The experiments of the Friedman test with Bonferroni–Dunn procedure are divided into two parts corresponding to the two groups of parameters, and the results are shown in Tables S.XIII and S.XIV in the supplementary material, respectively. The p-value shows the result of Bonferroni–Dunn procedure. "Y" represents the corresponding setting performs significantly different from the adopted setting at the significance level $\alpha = 0.05$, while "N" represents there is no significant difference. For the first group of parameters, the performance of the adopted setting is not significantly different from 15, 14, and 16 settings out of all the 25 settings at the three accuracy levels, respectively. For the second group of parameters, the performance of the adopted setting is not significantly different from 17, 17, and 16 settings out of all the 25 settings at the three accuracy levels, respectively. The above results show that the parameter setting has an influence on the performance of DIDE, but it is not highly sensitive.

In addition, we further analyze three parameters in detail to show their contributions to the algorithm, i.e., the population size N, the access threshold at, and the maximal consecutive generations mcg.

DIDE sets 100 as the population size, denoted as "N = 100," while "N = 50," "N = 200," and "N = 300" are further tested for comparison. The experimental results of PR at the accuracy level ε = 1.0E–04 are shown in Table IV. The best results on each function are bolded. Compared with the other settings, the numbers of functions that N = 100

TABLE IV
EXPERIMENTAL RESULTS OF PR IN THE PARAMETER ANALYSIS OF THE POPULATION SIZE AT THE ACCURACY LEVEL $\varepsilon = 1.0\text{E}{-}04$

Func	N=100	N=50	N=200	N=300
F_1	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**(≈)
F_2	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**(≈)
F_3	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**(≈)
F_4	**1.000**	**1.000**(≈)	**1.000**(≈)	0.790(+)
F_5	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**(≈)
F_6	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**(≈)
F_7	**0.921**	0.908(≈)	0.894(+)	0.895(+)
F_8	0.692	0.567(+)	0.759(−)	**0.764**(−)
F_9	**0.571**	0.543(+)	0.570(≈)	0.566(≈)
F_{10}	**1.000**	**1.000**(≈)	**1.000**(≈)	**1.000**(≈)
F_{11}	**1.000**	**1.000**(≈)	0.987(+)	0.960(+)
F_{12}	**1.000**	0.995(≈)	0.993(≈)	0.953(+)
F_{13}	**0.987**	0.980(≈)	0.950(+)	0.863(+)
F_{14}	**0.773**	0.737(+)	0.703(+)	0.703(+)
F_{15}	**0.748**	0.693(+)	0.740(≈)	0.735(≈)
F_{16}	**0.667**	**0.667**(≈)	**0.667**(≈)	**0.667**(≈)
F_{17}	**0.593**	0.535(+)	0.578(≈)	0.573(≈)
F_{18}	**0.667**	0.650(+)	0.660(≈)	0.590(+)
F_{19}	**0.543**	0.495(+)	0.515(+)	0.358(+)
F_{20}	**0.355**	0.315(+)	0.000(+)	0.000(+)
+		8	6	9
≈		12	13	10
−		0	1	1

performs significantly better, not significantly different, and significantly worse, are counted and presented in the bottom of each column. $N = 100$ outperforms $N = 50$ on eight functions, i.e., F_8, F_9, F_{14}, F_{15}, and $F_{17}-F_{20}$, while their performances are not significantly different on the other functions. $N = 200$ and $N = 300$ have better performance than $N = 100$ on F_8. However, they perform very poorly on F_{20}. The complete experimental results of PR at all three accuracy levels (1.0E–03, 1.0E–04, and 1.0E–05) are shown in Table S.XV in the supplementary material. We can see that the adopted setting $N = 100$ obtains the overall better results.

In DIDE, the access threshold *at* in lifetime mechanism is set to 0.8. The other three settings, "$at = 0.2$," "$at = 0.5$," and "$at = 1.0$," are tested for comparison. Notice that "$at = 1.0$" means all life-exhausted individuals can access the archive. F_8, F_{11}, and F_{13} are three typical functions to illustrate the effect of *at* and the experimental results of PR on these three functions at the accuracy level $\varepsilon = 1.0\text{E}{-}04$ are shown in Fig. 3. A small *at* is preferred on F_8. As shown in Fig. 2(a), F_6's global peaks are significantly higher than the local peaks. F_8 is the 3-D version of F_6 that has the same characteristic. Therefore, on F_8, the fitness value of the individual around a global peak is significantly higher than the one around a local peak. In this case, a small *at* can successfully filter those life-exhausted individuals around the local peaks while preserving those around the global peaks. However, the performance of a large *at* is better on F_{11} and F_{13}. From Fig. 2(c) and (d), we can see that these two functions have some peaks that

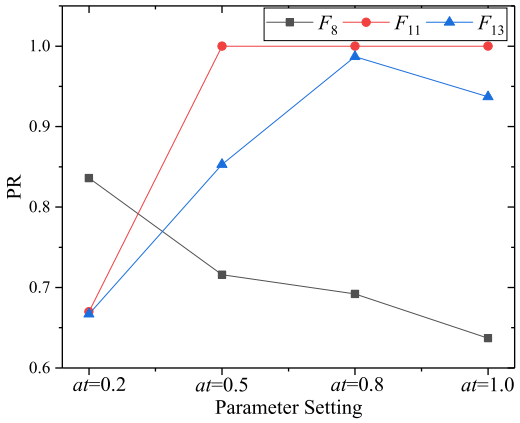


Fig. 3. Experimental results of four different *at* settings on F_8, F_{11}, and F_{13} at the accuracy level $\varepsilon = 1.0\text{E}{-}04$.

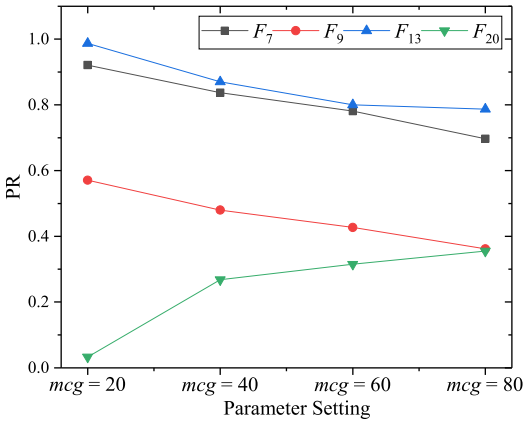


Fig. 4. Experimental results of four different *mcg* settings on F_7, F_9, F_{13}, and F_{20} at the accuracy level $\varepsilon = 1.0\text{E}{-}04$.

are "smooth" and have a large region, which are marked with black circles. In addition, there are also some peaks that are sharp and have a small region, which are marked with red stars. Those individuals in the region of the smooth peaks can easily have high fitness value. However, an individual that is already very close to a sharp peak still has a small fitness value. Therefore, a large *at* in this case is more likely to preserve the life-exhausted individuals around the sharp peaks. The complete experimental results of these four *at* settings on all 20 benchmark functions are shown in Table S.XVI in the supplementary material. The best results on each function are bolded. We can see that the adopted setting "$at = 0.8$" obtains a good tradeoff between filtering the life-exhausted individuals around local peaks and preserving those around global peaks, which performs better than the other settings.

The maximal consecutive generations *mcg* in the ARA strategy is adaptively adjusted according to (3) in DIDE. The higher the dimension size of the benchmark function, the larger *mcg* is. To verify the effectiveness of the adaptive *mcg*, four fixed settings, "$mcg = 20$," "$mcg = 40$," "$mcg = 60$," and "$mcg = 80$" are tested. The experimental results on four typical functions, F_7, F_9, F_{13}, and F_{20} at the accuracy level $\varepsilon = 1.0\text{E}{-}04$ are shown in Fig. 4. According to (3), F_7, F_9, and F_{13} are 2-D or 3-D functions where a small *mcg* (i.e., 20)

is adopted in DIDE, while F_{20} is a 20-D function where a large *mcg* (i.e., 80) is adopted. From Fig. 4, we can see that a small *mcg* performs better on F_7, F_9, and F_{13} while a large *mcg* is preferred on F_{20}. The complete experimental results of the four *mcg* settings on all 20 benchmark functions are shown in Table S.XVII in the supplementary material. On each function, the best results are bolded and the adopted *mcg* setting in DIDE plays as the baseline for the Wilcoxon rank-sum test. From the table, DIDE's adopted *mcg* settings have the best performance on all functions, which verifies the effectiveness of the adaptive *mcg* in DIDE.

V. Conclusion

In this article, a novel DIDE algorithm is proposed for MMOPs. DIDE is based on the DIMP framework that lets each individual act as a distributed unit to track a peak. The mutation operation in DIDE is conducted according to the virtual population. The ARA strategy is proposed to control the construction of virtual population, helping each individual explore the search space sufficiently to locate a peak and then gradually approach it. In addition, the lifetime mechanism and the ELM are incorporated into DIDE. On the one hand, the lifetime mechanism enhances the population diversity to locate more peaks. On the other hand, the ELM refines the solution accuracy on the found peaks. Experiments are conducted on 20 multimodal test functions in the CEC'2013 benchmark set. The experimental results show that the overall performance of DIDE is better than the other 13 state-of-the-art multimodal optimization algorithms. In the future, we will extend DIDE to deal with some MMOPs in real-world applications, such as finding multiple Nash equilibria in electricity markets [1] and the multimodal design optimization of permanent magnet synchronous machine [2]. Moreover, we will test DIDE's performance in solving other kinds of problems such as multiobjective optimization problems [43]–[45].

References

[1] F. Zaman, S. M. Elsayed, T. Ray, and R. A. Sarkerr, "Evolutionary algorithms for finding Nash equilibria in electricity markets," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 536–549, Aug. 2018.

[2] B. D. S. G. Vidanalage, M. S. Toulabi, and S. Filizadeh, "Multimodal design optimization of V-shaped magnet IPM synchronous machines," *IEEE Trans. Power Electron.*, vol. 33, no. 3, pp. 1547–1556, Sep. 2018.

[3] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 858–869, Oct. 2019. doi: 10.1109/TEVC.2019.2893614.

[4] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[5] Y. Yu, S. Gao, Y. Wang, J. Cheng, and Y. Todo, "ASBSO: An improved brain storm optimization with flexible search length and memory-based selection," *IEEE Access*, vol. 6, pp. 36977–36994, 2018.

[6] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht, "Seeking multiple solutions: An updated survey on niching methods and their applications," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 518–538, Aug. 2017.

[7] X.-F. Liu *et al.*, "Historical and heuristic-based adaptive differential evolution," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2018.2855155.

[8] Z.-H. Zhan *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[9] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 1382–1389.

[10] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," in *Proc. Conf. Genet. Evol. Comput.*, 2005, pp. 873–880.

[11] B. Y. Qu, P. N. Suganthan, and J. J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 601–614, Oct. 2012.

[12] W. Gao, G. G. Yen, and S. Liu, "A cluster-based differential evolution with self-adaptive strategy for multimodal optimization," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1314–1327, Aug. 2014.

[13] S. Hui and P. N. Suganthan, "Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 64–74, Jan. 2016.

[14] S. Biswas, S. Kundu, and S. Das, "An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1726–1737, Oct. 2014.

[15] S. Biswas, S. Kundu, and S. Das, "Inducing niching behavior in differential evolution through local information sharing," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 246–263, Apr. 2015.

[16] Z.-J. Wang *et al.*, "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 894–908, Dec. 2018.

[17] T. C. Goldsmith, "Aging as an evolved characteristic—Weismann's theory reconsidered," *Med. Hypotheses*, vol. 62, no. 2, pp. 304–308, 2004.

[18] T. C. Goldsmith, *The Evolution of Aging*. Crownsville, MD, USA: Azinet Press, 2013.

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

[20] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evol. Comput.*, vol. 10, no. 3, pp. 207–234, 2002.

[21] Z.-J. Wang *et al.*, "Automatic niching differential evolution with contour prediction approach for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, to be published. doi: 10.1109/TEVC.2019.2910721.

[22] C. Stoean, M. Preuss, R. Stoean, and D. Dumitrescu, "Multimodal optimization by means of a topological species conservation algorithm," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 842–864, Dec. 2010.

[23] L. Li and K. Tang, "History-based topological speciation for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 136–150, Feb. 2015.

[24] H. Zhao *et al.*, "Local binary pattern based adaptive differential evolution for multimodal optimization problems," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2927780.

[25] B. Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 387–402, Jun. 2013.

[26] Z. Wang, Q. Zhang, A. Zhou, M. Gong, and L. Jiao, "Adaptive replacement strategies for MOEA/D," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 474–486, Feb. 2016.

[27] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[28] M. Gong, H. Li, D. Meng, Q. Miao, and J. Liu, "Decomposition-based evolutionary multiobjective optimization to self-paced learning," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 288–302, Apr. 2019.

[29] J. Sun, S. Gao, H. Dai, J. Cheng, M. Zhou, and J. Wang, "Bi-objective elite differential evolution algorithm for multivalued logic networks," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2018.2868493.

[30] X.-F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587–602, Aug. 2019.

[31] J. Yao, N. Kharma, and P. Grogono, "Bi-objective multipopulation genetic algorithm for multimodal function optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 80–102, Feb. 2010.

[32] K. Deb and A. Saha, "Multimodal optimization using a bi-objective evolutionary algorithm," *Evol. Comput.*, vol. 20, no. 1, pp. 27–62, Mar. 2012.

[33] A. Basak, S. Das, and K. C. Tan, "Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection," *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 666–685, Oct. 2013.
[34] R. Cheng, M. Li, K. Li, and X. Yao, "Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape approximation and peak detection," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 692–706, Oct. 2018.
[35] Y. Wang, H.-X. Li, G. G. Yen, and W. Song, "MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 830–843, Apr. 2015.
[36] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 1, pp. 32–40, Jan. 1975.
[37] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, Aug. 1995.
[38] X. Li, A. Engelbrecht, and M. G. Epitropakis, "Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization," Evol. Comput. Mach. Learn. Group, RMIT Univ., Melbourne, VIC, Australia, Rep., 2013. [Online]. Available: http://titan.csit.rmit.edu.au/∼e46507/cec13-niching/competition/cec2013-niching-benchmark-tech-report.pdf
[39] J. E. Fieldsend, "Running up those hills: Multi-modal search with the niching migratory multi-swarm optimiser," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 2593–2600.
[40] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–614, Feb. 2019.
[41] G. Taguchi, R. Jugulum, and S. Taguchi, *Computer-Based Robust Engineering: Essentials for DFSS*. Milwaukee, WI, USA: ASQ Qual. Press, 2004.
[42] J. Ji, S. Gao, S. Wang, Y. Tang, H. Yu, and Y. Todo, "Self-adaptive gravitational search algorithm with a modified chaotic local search," *IEEE Access*, vol. 5, pp. 17881–17895, 2017.
[43] Z. Wang, Y. Ong, and H. Ishibuchi, "On scalable multiobjective test problems with hardly dominated boundaries," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 217–231, Apr. 2019.
[44] M. Gong, X. Jiang, H. Li, and K. C. Tan, "Multiobjective sparse nonnegative matrix factorization," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2941–2954, Aug. 2019.
[45] Z. Wang, Y.-W. Ong, J. Sun, A. Gupta, and Q. Zhang, "A generator for multiobjective test problems with difficult-to-approximate Pareto front boundaries," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 556–571, Aug. 2019.

Zong-Gan Chen (S'17) received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2016. He is currently pursuing the Ph.D. degree in computer science and technology with the South China University of Technology, Guangzhou.

His current research interests include ant colony optimization, differential evolution, and their applications in real-world optimization problems.

Zhi-Hui Zhan (M'13–SM'18) received the bachelor's and Ph.D. degrees in computer science from the Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is also the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems, and in environments of cloud computing and big data.

Prof. Zhan was a recipient of the Outstanding Youth Science Foundation from National Natural Science Foundations of China in 2018 and the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. He was awarded the China Computer Federation Outstanding Ph.D. Dissertation for his doctoral dissertation and the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of the *Neurocomputing* and the *International Journal of Swarm Intelligence Research*.

Hua Wang received the Ph.D. degree from the University of Southern Queensland, Toowoomba, QLD, Australia, in 2004.

He was a Professor with the University of Southern Queensland. He is currently a Full Time Professor with Victoria University, Melbourne, VIC, Australia. He has over ten years teaching and working experience in applied informatics at both enterprise and university. He has expertise in electronic commerce, business process modeling, and enterprise architecture. As a Chief Investigator, three Australian Research Council Discovery grants have been awarded since 2006, and 200 peer reviewed scholar papers have been published. Six Ph.D. students have already graduated under his principal supervision.

Jun Zhang (F'17) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.