VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

Local binary pattern-based adaptive differential evolution for multimodal optimization problems

This is the Published version of the following publication

Zhao, Hong, Zhan, Zhi-Hui, Lin, Ying, Chen, Xiaofeng, Luo, X, Zhang, Jie, Kwong, Sam and Zhang, Jun (2019) Local binary pattern-based adaptive differential evolution for multimodal optimization problems. IEEE Transactions on Cybernetics, 50 (7). pp. 3343-3357. ISSN 2168-2267

The publisher's official version can be found at
https://ieeexplore.ieee.org/document/8792370
Note that access to this version may require subscription.

Downloaded from VU Research Repository https://vuir.vu.edu.au/45250/

Local Binary Pattern-Based Adaptive Differential Evolution for Multimodal Optimization Problems

Hong Zhao, *Student Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, Ying Lin, *Member, IEEE*, Xiaofeng Chen, Xiao-Nan Luo, Jie Zhang, Sam Kwong, *Fellow, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The multimodal optimization problem (MMOP) requires the algorithm to find multiple global optima of the problem simultaneously. In order to solve MMOP efficiently, a novel differential evolution (DE) algorithm based on the local binary pattern (LBP) is proposed in this paper. The LBP makes use of the neighbors' information for extracting relevant pattern information, so as to identify the multiple regions of interests, which is similar to finding multiple peaks in MMOP. Inspired by the principle of LBP, this paper proposes an LBP-based adaptive DE (LBPADE) algorithm. It enables the LBP operator to form multiple niches, and further to locate multiple peak regions in MMOP. Moreover, based on the LBP niching information, we develop a niching and global interaction (NGI) mutation strategy and an adaptive parameter strategy (APS) to fully search the niching areas and maintain multiple peak regions. The proposed NGI mutation strategy incorporates information from both the niching and the global areas for effective exploration, while APS adjusts the parameters of each individual based on its own LBP information and guides the individual to the promising direction. The proposed LBPADEalgorithm is evaluated on the extensive MMOPs test functions. The experimental results show that LBPADE outperforms or at least remains competitive with some state-of-the-art algorithms.**

***Index Terms*—Adaptive differential evolution (DE), DE, local binary pattern (LBP) strategy, multimodal optimization problems (MMOPs).**

I. Introduction

Differential evolution (DE) is a kind of evolutionary algorithm (EAs) proposed by Storn and Price in 1995 [1]. Like other EAs, DE is a class of heuristic optimization algorithms that includes mutation, crossover, and selection operators. The DE algorithm is efficient for solving many real-world optimization problems [2]–[5].

With the increasing complexity of the real-world problems, many problems have multiple optimal solutions, namely, multimodal optimization problems (MMOPs). Over the past decades, MMOPs have drawn considerable attention [6]. Many researchers try to use EAs [7] and swarm intelligence algorithms [8] to solve MMOPs, such as the genetic algorithm [9], ant colony optimization [10], estimation of distribution algorithm [11], particle swarm optimization (PSO) [12], and DE [13]. Although tremendous efforts have been put into utilizing the above algorithms to solve MMOPs, there are still many drawbacks as discussed in [14]. First, how to efficiently form niches to locate as many peaks as possible is still a challenge. Second, the parameters of the algorithm are still difficult to set for balancing the exploration and exploitation in solving MMOPs. Third, it is still difficult to maintain the found optima until the end of the evolution.

When dealing with the above difficulties, the existing methods have the following limits. First, some of the existing methods try to divide the population into several separate niches. However, it is difficult to determine the niche parameters, such as the number of niches and their sizes. Second, if we use only the local information from the niche in the evolutionary operator to guide the individuals, the algorithms may get trapped in local optima. This problem would be more serious especially by the influence of the structural bias of the algorithms [15], [16]. Third, some of the existing methods adopt fixed parameters during the entire evolution, which is not efficient to balance the exploration and exploitation abilities in solving MMOPs. Although self-adaptive parameters may sometimes tend to have weak exploration ability and are

Manuscript received January 28, 2019; revised May 14, 2019; accepted June 30, 2019. Date of publication August 8, 2019; date of current version June 16, 2020. This work was supported in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundation of China under Grant 61772207 and Grant 61873097, in part by the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars under Grant 2014A030306038, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, in part by the Guangdong–Hong Kong Joint Innovation Platform under Grant 2018B050502006, and in part by the Hong Kong GRF-RGC General Research Fund 9042489 under Grant CityU 11206317. This paper was recommended by Associate Editor H. Takagi. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

H. Zhao 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).

Y. Lin is with the Department of Psychology, Sun Yat-sen University, Guangzhou 510006, China.

X. Chen is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710126, China.

X.-N. Luo is with the Guangxi Key Laboratory of Image and Graphic Intelligent Processing, Guilin University of Electronic Technology, Guilin 541004, China.

J. Zhang is with the School of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China.

S. Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

J. Zhang is with Victoria University, Melbourne, VIC 8001, Australia (e-mail: csjun@scut.edu.cn).

This paper 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 paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2927780

2168-2267 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

more prone to premature convergence [17], they can be efficient if the diversity is well addressed by cooperating with the niche information. Therefore, there is a great need to design an efficient niching method that can form niches without sensitive parameters to find as many peaks as possible. Moreover, an efficient evolutionary operator and an adaptive parameter control strategy that cooperate with the niche strategy are in great need to balance the exploration and exploitation abilities to refine the found peaks and to maintain them during the entire evolution process.

To these aims, this paper borrows the idea of the local binary pattern (LBP) operator from image processing to the optimization domain for solving MMOPs efficiently. Similar to the multiple peaks detection in MMOPs, researchers also want to find all of the regions of interest in a picture in the image processing. LBP is a simple yet efficient multiresolution approach used in the image processing [18], which can process the grayscale and rotation-invariant texture classification based on the nonparametric discrimination of prototype distributions. Meanwhile, LBP uses the local pattern information to identify multiple textures in order to identify multiple objects, which can be analogous to locate multiple peaks in MMOPs. Therefore, by borrowing this idea, a novel LBP-based niching strategy is proposed, which also uses the local information of each individual to help the individual construct niche. In this way, more peak regions can be detected because all of the individuals can form their own LBP niches. Moreover, the current individual can be guided by both the local information from niche and the global information from the population, in order to prevent individuals from being trapped into local optima. Furthermore, we can adaptively control the relevant parameters according to the neighbors' information from the LBP niche to reduce the limitation of fixed parameters.

Inspired by the aforementioned motivations, we incorporate the LBP-based niching strategy in DE and propose an LBP-based adaptive DE (LBPADE) in this paper. The proposed LBPADE algorithm has the following three advantages.

1) A novel niching strategy based on LBP is designed to identify as many peaks as possible. This strategy avoids the sensitive niching parameters, such as the number of niches and their sizes.
2) A new mutation strategy called the niching and global interaction (NGI) mutation strategy is developed to guide the individual to a more promising position by using both the local information of the LBP niche and the global information of the entire population.
3) Instead of using fixed parameters in DE, this paper proposes a new adaptive parameter strategy (APS) for each individual according to its fitness and distribution information about the LBP niche, to relieve the sensitivity of parameters like scaling factor F and crossover rate CR.

The LBP-based niching strategy can help the LBPADE to locate as many peaks as possible, while the NGI and APS based on the LBP niche can help the LBPADE to balance the exploration and exploitation abilities efficiently. The numerical experiments are conducted on all 20 widely used multimodal benchmark functions in CEC'2013. The experimental results show that the LBPADE is superior to other algorithms in comparison.

The remainder of this paper is structured as follows. Section II describes the basic DE algorithm and the related works on MMOPs. Section III presents the details of the proposed LBPADE. Next, Section IV shows the extensive experimental studies. Finally, Section V draws the conclusion of this paper.

II. Related Works

A. DE

The basic idea of DE is to generate new individuals through the difference between individuals and select the better individual to enter the next generation. The standard DE process includes the following steps.

1) Initialization: The initial population is randomly generated within a given boundary domain as

$$x_{i,j} = L_j + \text{rand}(0, 1) \times \left(U_j - L_j\right) \tag{1}$$

where $i = 1, 2, \ldots, NP$ and $j = 1, 2, \ldots, D$. Herein, NP represents the population size, and D is the problem dimension. rand(0, 1) is a random number uniformly distributed in the interval of (0, 1), and L_j and U_j denote the lower and upper bounds of the jth dimension, respectively.

2) Mutation Operator: At each generation, a mutation vector $\boldsymbol{v}_i$ is obtained based on the difference between individuals. Here, we list some typical mutation strategies as follows.

DE/rand/1:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F \times \left(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}\right). \tag{2}$$

DE/best/1:

$$\boldsymbol{v}_i = \boldsymbol{x}_{\text{best}} + F \times \left(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}\right). \tag{3}$$

DE/current-to-best/1:

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F \times (\boldsymbol{x}_{\text{best}} - \boldsymbol{x}_i) + F \times (\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}). \tag{4}$$

DE/current-to-rand/1:

$$\boldsymbol{v}_i = \boldsymbol{x}_i + \text{rand}(0, 1) \times (\boldsymbol{x}_{r1} - \boldsymbol{x}_i) + F \times (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}). \tag{5}$$

DE/current-to-pbest/1 (without archive):

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F \times \left(\boldsymbol{x}^p_{\text{best}} - \boldsymbol{x}_i\right) + F \times (\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}). \tag{6}$$

DE/current-to-pbest/1 (with archive):

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F \times \left(\boldsymbol{x}^p_{\text{best}} - \boldsymbol{x}_i\right) + F \times \left(\boldsymbol{x}_{r1} - \boldsymbol{x}^*_{r2}\right) \tag{7}$$

where $\boldsymbol{v}_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,D}], i \neq r_1 \neq r_2 \neq r_3$, r_1, r_2, $r_3 \in \{1, 2, \ldots, NP\}$, and F is the scaling factor. $\boldsymbol{x}_{\text{best}}$ is the best individual that has the best fitness value in the population. $\boldsymbol{x}^{\boldsymbol{p}}_{\text{best}}$ is randomly chosen as one of the top $100p\%$ individuals in the current population with $p \in (0, 1)$. $\boldsymbol{x}^*_{r2}$ is randomly chosen from the union of $\boldsymbol{P}$ and $\boldsymbol{A}$, where $\boldsymbol{P}$ is the set of current population and $\boldsymbol{A}$ is the set of archived inferior solutions [19].

3) Crossover Operator: After the mutation operation, the crossover operator is performed on the individuals $\boldsymbol{v_i}$ and $\boldsymbol{x_i}$ to form a trial vector $\boldsymbol{u}_i = [u_{i,1}, u_{i,2}, \ldots, u_{i,D}]$. Herein, we describe two typical crossover operators, namely, the binomial crossover and exponential crossover. The binomial crossover is used in this paper.

In binomial crossover, each dimension of $\boldsymbol{u}_i$ is separately determined to come from $\boldsymbol{v_i}$ or $\boldsymbol{x_i}$ by a crossover rate CR as

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } \text{rand}(0,1) \leq CR \text{ or } j = jrand \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{8}$$

where rand(0, 1) returns a random number between 0 and 1, while the *jrand* is a random index in $\{1, 2, \ldots, D\}$ to ensure that at least one dimension of $\boldsymbol{u}_i$ comes from $\boldsymbol{v}_i$.

In exponential crossover, L consecutive dimensions come from $\boldsymbol{v}_i$ as

$$u_{i,j} = \begin{cases} v_{i,j}, & \forall j \in \{k, \langle k+1 \rangle_D, \ldots, \langle k+L-1 \rangle_D\} \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{9}$$

where dimension k is randomly selected from $\{1, 2, \ldots, D\}$. Then, all of the following L dimensions come from $\boldsymbol{v_i}$. Note that $<\cdot>_D$ means modulo D and returns D if the result is 0. Moreover, L is the length of the sequence, which is no longer than D and is determined by the crossover rate CR as shown in Algorithm S.I in the supplementary material.

4) Selection Operator: The selection operator is conducted by comparing the objective values of the original individual $\boldsymbol{x}_i$ and the trial vector $\boldsymbol{u}_i$ using (10) for a maximization problem, where the better one is selected for the next generation

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{u}_i, & \text{if } f(\boldsymbol{u}_i) \geq f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i, & \text{otherwise.} \end{cases} \tag{10}$$

The DE repeats the above mutation, crossover, and selection operators until it meets the terminal conditions. The pseudocode of a traditional DE variant called DE/rand/1/bin is given in Algorithm S.II in the supplementary material.

B. Related Works on MMOPs

The algorithm for solving MMOPs is required to maintain the diversity of population to find as many peaks as possible. Moreover, due to the limited budget on the fitness evaluations (FEs), the algorithm is also required to converge fast in each peak region. To better review the related works on these efforts to solve MMOPs, we attempt to describe them in three aspects.

1) Niching Strategies: In order to localize as many peaks as possible, a fruitful research line is based on the usage of niching schemes [20], [21]. The two most famous niching methods are the crowding method [20] and the speciation method [21]. The crowding DE (CDE) compares the fitness of an offspring individual with the nearest parental individual of the crowd formed by some parental individuals. The offspring will replace its nearest parental individual if it has a better fitness value. Otherwise, the offspring will be ignored. The speciation DE (SDE) solves MMOPs by evolving multiple species, each of which evolves independently around a peak region to locate more global optima. However, both of the two niching methods introduce additional parameters, that is, the crowding size in crowding and the species radius in speciation, which are problem-dependent and highly sensitive to the algorithm performance.

To reduce the influence of parameters in niching methods, some improved niching methods have been proposed. Li [22] proposed using parameter-free ring topology according to the index of the individual for niching, resulting in R2PSO and R3PSO. Gao *et al.* [23] introduced a self-adaptive cluster-based DE (Self-CCDE) for MMOPs. It adopted the multipopulation strategy to locate different optima, and employed the self-adaptive parameter control to enhance searchability

Therefore, designing an efficient niching method that can form niches without sensitive parameters is in great need. To this aim, this paper proposes an LBP-based niching method where the neighbors that form the niche can be determined similar to the idea of the LBP operator in image processing. Moreover, our proposed APS based on the LBP information can be adaptive control of the parameters in DE. The proposed LBPADE algorithm will be compared with the above algorithms with different niching strategies.

2) New Evolutionary Operators: Since MMOP requires the algorithm to find all of the peaks and refine their accuracy, the population diversity and convergence ability are both very important. Therefore, many new evolutionary operators have been proposed to combine with the niching strategy. Using the clustering methods to initialize population can increase the diversity [24] and may be good in dealing with MMOPs. Qu *et al.* [25] proposed a neighborhood mutation strategy to design the neighborhood CDE (NCDE) and neighborhood SDE (NSDE) algorithms. These algorithms are promising in maintaining the multiple optima found during evolution. Qu *et al.* [26] also proposed a distance-based locally informed particle swarm (LIPS) algorithm, which utilized several local bests to guide the evolution of particles. Biswas *et al.* [27] introduced an improved parent-centric normalized neighborhood mutation operator for DE (PNPCDE), which contained a niching scheme by combining the parent-centric mutation operator with the crowding replacement rule. They also introduced an information-sharing mechanism among the individuals to enhance the niching behavior and proposed the LoICDE algorithm [28].

Therefore, how to design an efficient evolutionary operator to balance the population diversity and convergence ability is significant in solving MMOP. To this aim, this paper proposes the NGI mutation strategy that can combine the niche information and global information to balance the exploration (diversity) and exploitation (convergence) abilities of the algorithm. The proposed LBPADE algorithm will be compared with the above algorithms with different evolutionary operators.

3) Multiobjective Techniques: In recent years, some researchers also transformed MMOPs into multiobjective optimization problems [29]–[31]. Generally, the first objective can be the multimodal function itself and the second objective is a specially designed function. Cheng *et al.* [32] proposed an estimation of the potential optimal areas method, which utilized an adaptive diversity indicator as the second optimization

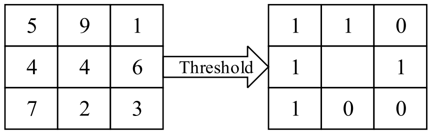

Fig. 1. Process of LBP in image processing.

objective. Basak *et al.* [33] defined mean distance-based selection as the second objective. Differently, a very efficient way was proposed by Wang *et al.* [34] where the MMOP was transformed into a multiple objective optimization problem whose two objectives are designed according to the definition of multiobjective optimization, so that the two objectives conflict. This is the MOMMOP algorithm and shows general better performance than other multiobjective-based methods. Therefore, the MOMMOP algorithm will be adopted to compare with our proposed LBPADE algorithm.

III. LBPADE

In this section, we first introduce the detailed process of the niching strategy based on LBP. Next, we describe the NGI mutation strategy, the new APS based on LBP, and a boundary constraint strategy (BCS). Finally, the complete LBPADE algorithm is derived.

A. LBP Operator in Image Processing

LBP is widely used to extract the local texture feature of the image in image processing [18]. More specifically, LBP marks the difference between the center-point pixel and its neighborhood pixel by a threshold value, where the neighborhood is defined as a window of 3×3. Therefore, a local area of LBP has nine pixels. LBP uses the gray value of the window center as a threshold, and compares the gray value of the center with the other eight adjacent pixels. As shown in Fig. 1, if the gray value of the surrounding pixel is not smaller than that of the center pixel, the pixel is marked as 1; otherwise 0. The LBP in the window is used to reflect the texture information of this local area. Using the LBP operator, we can find the edge of the image in the local area, so we can identify different objects in an image.

Similar to the idea of LBP in image processing, we can also describe the information of the neighbors around each individual in DE when solving MMOPs. That is, we regard each individual as a pixel. Each individual (pixel), together with its similar M individuals (M nearest surrounding pixels), forms a niche (local area). Herein, the similarity is measured by the Euclidean distance to simulate the surrounding information, and M can be set as 8 due to the original LBP being within a 3×3 window with 8 surrounding pixels. We also regard the fitness value of each individual as the gray value of each pixel. Then, we can compare the fitness value of each individual in the current niche with the fitness value of the current (center) individual. For example, for maximization MMOPs, if a neighboring individual has a fitness value larger than or equal to the current individual (i.e., the neighboring individual has a better fitness value), it is marked with 1. Otherwise, it is marked with 0. Then, we introduce an external archive set $\boldsymbol{S}$ to store all of the individuals marked with 1 in the current niching. Note that each individual forms its own LBP-based niche and has its own set $\boldsymbol{S}$ to store the individuals in the current niche that are equal or better than itself. In this way, the current individual can learn from the better individuals in $\boldsymbol{S}$ via a novel-designed NGI mutation strategy, which will be discussed in the following section.

B. NGI Mutation Strategy

In classic DE, the differential vector(s) in the mutation operator are generated by individuals randomly selected from the entire population, without concerning the distance among them. Even individuals far from each other can be used to produce the differential vector(s). Such a mutation operator might not be suitable for MMOPs, because it may slow down the convergence toward the global optima in each peak region, despite not increasing the overall population diversity. In many DE algorithms specifically designed for MMOPs, the mutation strategy is modified to make use of the niching information [20], [21], that is, the differential vector(s) in mutation are generated by individuals from the same niche. These methods are helpful in locating more peaks, but may also increase the risk of getting trapped into local optima due to the potential loss of population diversity.

In order to accurately locate the regions with optimal solutions and maintain global searchability, an NGI mutation strategy is proposed, which combines the local information of the niche and the global information of the entire population. This strategy not only increases the diversity of the population but also enhances the convergence of the population. That is, for an individual $\boldsymbol{x}_i$, the NGI mutation strategy is as

$$v_i = \begin{cases} \boldsymbol{x}_i + F_i \times (\boldsymbol{x}_{n\text{best}} - \boldsymbol{x}_i) + F_i \times \left(\boldsymbol{x}_{g\_r_1} - \boldsymbol{x}_{g\_r_2}\right) \\ \qquad \text{if } |\boldsymbol{S}| \geq 1,\, g\_r_1 \neq g\_r_2 \neq i \in \boldsymbol{N} \\ \boldsymbol{x}_i + F_i \times \left(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}\right) \\ \qquad \text{otherwise } |\boldsymbol{S}| = 0,\;\; r_1 \neq r_2 \neq i \in \boldsymbol{M} \end{cases} \tag{11}$$

where $\boldsymbol{M}$ is the set that stores the M neighbors, $\boldsymbol{N}$ is the set that stores the entire population, and $|\boldsymbol{N}| = NP$. The set $\boldsymbol{S}$ stores the individuals that have an equal or better fitness value than the value of $f(\boldsymbol{x}_i)$, that is, $\boldsymbol{S} = \{\boldsymbol{x}_j | f(\boldsymbol{x}_j) \geq f(\boldsymbol{x}_i)\}$, $\boldsymbol{x}_j \in \boldsymbol{M}$. r_1 and r_2 are randomly selected from the set $\boldsymbol{M}$, while g_r_1 and g_r_2 are randomly selected from the set $\boldsymbol{N}$, and $\boldsymbol{x}_{n\text{best}}$ is the best individual in the current niche.

In our proposed NGI mutation strategy, there are two situations. One situation is that $|\boldsymbol{S}| \geq 1$, meaning that better individual(s) exit in the niche, which can guide the current individual $\boldsymbol{x}_i$. In this case, $\boldsymbol{x}_{n\text{best}}$ is used to guide $\boldsymbol{x}_i$, so that the individual $\boldsymbol{x}_i$ can locate the potential optima quickly. Meanwhile, in order to prevent individuals from being trapped into local optima, the global disturbance is also added in the NGI mutation strategy, that is, g_r_1 and g_r_2 are randomly selected from the entire population. This situation is shown as the if statement of (11). The other situation is that $|\boldsymbol{S}| = 0$, meaning that the current individual $\boldsymbol{x}_i$ is the best in the LBP-based niche. In this case, $\boldsymbol{x}_i$ may be close to the optimum. Therefore, we randomly select two individuals from the set $\boldsymbol{M}$ to generate local exploitation information for $\boldsymbol{x}_i$. This way, we can improve the convergence speed to the current potential optimum.

C. LBP-Based APS

In this section, to balance the exploration and exploitation abilities, a method of APS is proposed. APS can adjust the parameters based on the information of the fitness and the LBP niche of each individual.

1) Adaptive Parameter F: According to the LBP-based niche, we utilize the neighbor's information of each individual to find the promising direction of evolution in MMOPs. Generally speaking, when $|\boldsymbol{S}|$ is larger, the number of better individuals (the fitness is equal to or better than current individual $\boldsymbol{x}_i$) is large, and $\boldsymbol{x}_i$ needs to learn more from these better individuals, so F_i should be larger. On the contrary, if $|\boldsymbol{S}|$ is smaller, the fitness of the current individual is promising, and the number of better individuals is less, so $\boldsymbol{x}_i$ does not need too much learning, and F_i should be smaller. Therefore, the value of $|\boldsymbol{S}|/|\boldsymbol{M}|$ can be used to guide the learning scale (i.e., the parameter F_i) of the current individual.

Besides, it should be noticed that in the early stage of the evolution, most of the individuals are in exploration. Therefore, F_i should be larger to search more globally optimal regions. However, in the later stage of the evolution, most of the individuals are in exploitation. In order to accelerate the convergence speed, it is better to reduce F_i to avoid oscillation. Based on the above two considerations, F_i is set as

$$F_i = \begin{cases} \frac{|\boldsymbol{S}|}{|\boldsymbol{M}|} \times (b-a) + a, & \text{if } fe \le \lambda \cdot MaxFEs \\ \left[\frac{|\boldsymbol{S}|}{|\boldsymbol{M}|} \times (b-a) + a\right] \times 0.001, & \text{otherwise} \end{cases} \tag{12}$$

where a and b are the lower and upper boundaries of F_i, respectively; fe is the current number of FEs; $MaxFEs$ is the maximum number of FEs; and λ is a parameter that indicates the evolutionary stage of the algorithm. As the range of F is usually within [0.1, 0.9], a and b are simply set as 0.1 and 0.9, respectively, whose influences are also investigated in Table S.I in the supplementary material. Moreover, through the empirical studies, we can find that most of the individuals finish the exploration stage after 80% of the evolutionary process and start the exploitation stage in the last 20% process. Therefore, we set the value of λ as 0.8, which is investigated in Section IV-G. It should be noted that in the later stage of the evolution, most of the individuals are in the exploitation. In order to accelerate the convergence speed, it is better to reduce F_i to avoid oscillation. Therefore, we set F_i to shrink to 0.001 times in the later stage of the evolution, which helps to balance the exploration and exploitation of the population.

2) Adaptive Parameter CR: To better balance the diversity and convergence, the parameter CR_i of $\boldsymbol{x}_i$ is adaptively controlled based on the location distribution information of the individuals in its LBP niche. The detailed process of adaptively controlling CR_i is given in the following steps.

Step 1: According to the location of each individual $\boldsymbol{x}_j$ in the LBP niche, the center position of the $|\boldsymbol{M}|+1$ individuals is denoted as ***center***, whose kth dimension $center_k$ is

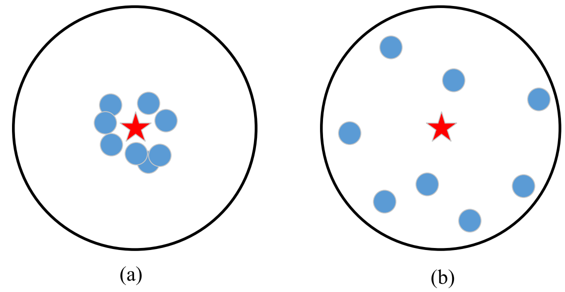

Fig. 2. Distribution of individuals in the current niche. (a) Concentrated and uniform. (b) Scattered and uneven.

calculated as

$$\text{center}_k = \frac{\sum_{j=1}^{|\boldsymbol{M}|+1} x_{j,k}}{|\boldsymbol{M}|+1} \tag{13}$$

where $k = 1, 2, \ldots, D$, and $x_{j,k}$ is the kth dimension of the jth individual in the current niching.

Step 2: Calculate the Euclidian distance d_j between the jth individual and the ***center*** as

$$d_j = \left\| \boldsymbol{center} - x_j \right\| \tag{14}$$

where $j = 1, 2, \ldots, |\boldsymbol{M}|+1$. The d_j reflects the distribution information of the individual $\boldsymbol{x}_i$ in the niche.

Step 3: Calculate the average distance $\bar{d}$ as

$$\bar{d} = \frac{\sum_{j=1}^{|\boldsymbol{M}|+1} d_j}{|\boldsymbol{M}|+1}. \tag{15}$$

The rationale of $\bar{d}$ is to reflect the uniformity of the individuals distribution in the current niche. As illustrated in Fig. 2(a), if all d_i have similar values, the individuals are likely to concentrate and distribute uniformly around the peak. Otherwise, the distribution of individuals in the current niche is likely to be scattered and uneven as in Fig. 2(b). In order to measure the uniformity (UN) of the niche, step 4 is designed.

Step 4: Calculate the UN of $\boldsymbol{x}_i$ in the current niche as

$$UN_i = \frac{\sum_{j=1}^{|\boldsymbol{M}|+1} \left(d_j - \bar{d}\right)^2}{|\boldsymbol{M}|+1}. \tag{16}$$

According to (16), the concentrated and uniform niche like Fig. 2(a) will have a small UN_i. Conversely, the UN_i will be large if the individuals of the current niche are more scattered and uneven as in Fig. 2(b).

Step 5: Adaptively control the value of CR_i according to UN_i as

$$CR_i = \left(1 - e^{-UN_i}\right) \times (n-m) + m \tag{17}$$

where m and n are similar as a and b in (12) to clamp the range of CR. Since the range of CR in DE is usually within [0.1, 0.9], the values of m and n are simply assigned to 0.1 and 0.9, respectively, whose influences are also investigated in Table S.II in the supplementary material.

Fig. S.I in the supplementary material shows the rationale and detailed changes of CR_i with different UN_i. It can be found that CR_i becomes larger as UN_i increases until CR_i reaches the maximum value of n, that is, 0.9. For a larger value of UN_i, the distribution of current niche is more uneven, indicating that the current individual has not reached the convergence stage, and needs to increase the search diversity. Therefore, a larger CR_i value is needed to obtain more information from the mutated vectors, which is helpful for increasing the diversity. On the contrary, when UN_i is smaller, the individuals' distribution of the niche is more concentrated and uniform. This indicates that the niche may have reached a convergence stage and, therefore, the value of CR_i should be smaller to make the individual keep more information from the current position, in order to increase the solution accuracy.

D. BCS

During evolution, the individual may exceed the search range due to the exploration in the mutation operator. A simple handle method is to directly set the exceeded dimension of the individual to the corresponding boundary value. This way may be helpful to increase the search diversity of the population in the early stage of the evolutionary process to find more peaks. However, in the later stage of the evolutionary process, most of the individuals have arrived convergence around the best individual of the niche. In this case, if the range-exceeded individuals are still set to the boundary, the convergence may be destroyed. Therefore, in order to balance the diversity and convergence, a novel BCS is proposed in this paper. The BCS deals with the range-exceeded individual according to different stages during the evolutionary process. In the early stage of the evolutionary process, the BCS resets the range-exceeded individual directly to the boundary (L_j or U_j). However, in the later stage of the evolutionary process, the BCS adopts another method to reset the range-exceeded individuals by using information of the best individual ($\boldsymbol{x}_{n\text{best}}$) in the current niching to accelerate the population convergence. In conclusion, when the solution $v_{i,j}$ is range exceeded, $v_{i,j}$ can be reset as

$$v_{i,j} = \begin{cases} U_j, & \text{if } \left(v_{i,j} > U_j\right) \text{ and } (fe < \mu \times MaxFEs) \\ L_j, & \text{if } \left(v_{i,j} < L_j\right) \text{ and } (fe < \mu \times MaxFEs) \\ x_{n\text{best},j}, & \text{otherwise} \end{cases} \tag{18}$$

where μ is a parameter that controls the way of dealing with boundary, and $\mu = 1\text{E-}04$, which is investigated in Section IV-F. $\boldsymbol{x}_{n\text{best}}$ is the best individual in the current niche.

E. Complete LBPADE Algorithm

This section provides the complete LBPADE algorithm. The initialization and the crossover operators are inherited from the traditional DE, and the NGI mutation operators have been introduced as above. Therefore, only the selection operator and the termination condition are described here.

1) Selection Operator: The selection operator is to find the nearest individual in the parental population to the current individual, and compare their fitness. Then, the individual with better fitness is selected for the next generation. For maximization MMOP, after forming the trial vector $\boldsymbol{u}_i$ by the crossover operator, the selection operator needs to find the nearest individual $\boldsymbol{x}_p$ to $\boldsymbol{u}_i$ in the parental population.

Algorithm 1 LBPADE

Begin
Random initialization population with size NP and set $fe = 0$;
While $fe < MaxFEs$ do
  For $i = 1$ to NP
    Find the most nearest (measured by Euclidean distance) $|\boldsymbol{M}|$ individuals of the current individual $\boldsymbol{x_i}$ to form a niche;
    Compute the mutation scaling factor F_i by Eq. (12) ;
    Produce a $\boldsymbol{v}_i$ using NGI mutation strategy by Eq. (11);
    Use Eq. (18) to reset $\boldsymbol{v}_i$ if the $\boldsymbol{v}_i$ is range-exceeded;
    Compute the crossover rate CR_i by Eq. (17);
    Produce a $\boldsymbol{u}_i$ using crossover operator by Eq. (8);
  End For
  For each trial vector $\boldsymbol{u}_i$
    Evaluate the fitness values of $\boldsymbol{u}_i$;
    Find the individual $\boldsymbol{x}_p$ that is nearest to $\boldsymbol{u}_i$ in parents population;
    Select $\boldsymbol{x}_i$ by comparing the fitness values of $\boldsymbol{u}_i$ and $\boldsymbol{x}_p$ as Eq. (19);
  End For
  $fe = fe + NP$;
End While
End

TABLE I
BASIC INFORMATION OF THE TEST FUNCTION

Function	MaxFEs	NP
F1-F5	5.00E+04	80
F6	2.00E+05	100
F7	2.00E+05	300
F8-F9	4.00E+05	300
F10	2.00E+05	100
F11-F13	2.00E+05	200
F14-F20	4.00E+05	200

The selection operator is represented as

$$\boldsymbol{x_i} = \begin{cases} \boldsymbol{u}_i, & \text{if } f(\boldsymbol{u}_i) \geq f\left(\boldsymbol{x}_p\right) \\ \boldsymbol{x}_p, & \text{otherwise.} \end{cases} \tag{19}$$

2) Termination Condition: The termination condition is defined by the maximum number of FEs ($MaxFEs$). If the termination criterion is satisfied, the algorithm stops and returns the fitness value of the final population as the result. Moreover, if the algorithm has found all known global peaks, it also stops.

Overall, the pseudocode of the complete LBPADE algorithm is outlined in Algorithm 1.

IV. EXPERIMENTAL VERIFICATION AND COMPARISON

A. Test Functions and Experimental Settings

The CEC'2013 benchmark set includes 20 multimodal test functions. All test functions are acquainted as maximization problems. The property of these multimodal test functions has been introduced in [35]. Basic information of the test functions is also summarized in Table I.

In order to validate the effectiveness of the proposed LBPADE algorithm, experimental tests on the benchmark functions are undertaken in this section. The performance of the proposed LBPADE algorithm will be compared with some state-of-the-art multimodal optimization algorithms. The related parameters used in this paper and the compared algorithms are as follows: the test function is the set of CEC'2013, and the other parameters are set as the original paper except for population size and $MaxFEs$.

Our proposed algorithm is compared with 11 state-of-the-art multimodal algorithms. They are CDE [20], SDE [21], R2PSO [22], R3PSO [22], Self_CCDE [23], NCDE [25], NSDE [25], LIPS [26], PNPCDE [27], LoICDE [28], and MOMMOP [34]. The reason for choosing these 11 algorithms is that they are representative algorithms in the three categories of related works on MMOP, as reviewed in Section II-B. Therefore, all 11 algorithms are adopted to be compared with LBPADE, where the first five algorithms (i.e., CDE, SDE, R2PSO, R3PSO, and Self-CCDE) are in the niching strategies category, the second five algorithms (i.e., NCDE, NSDE, LIPS, PNPCDE, and LoICDE) are in the new evolutionary operators category, and the last MOMMOP is the typical and well-performed algorithm in the multiobjective techniques category. All of these algorithms use the parameter settings recommended by their source literature, except for the population size (*NP*) and the termination criterion (*MaxFEs*). As listed in Table I, different settings of the population size and the termination criterion are adopted for different functions depending on their degrees of complexity, but the settings are set the same across all algorithms, and each algorithm is tested for 51 independent runs.

With the accuracy level ε set at 1E-04, the peak ratio (*PR*), the success rate (*SR*), and the average *FEs* (*AveFEs*) are calculated to evaluate the algorithmic performance. *PR* is defined as the average percentage of the global optima found in multiple runs

$$PR = \frac{\sum_{i=1}^{TR} HFP_i}{HKP \cdot TR} \tag{20}$$

where HFP_i is the number of have found peaks (*HFP*) in the end of the *i*th run, *HKP* is the number of have known peaks, and *TR* is the number of total runs. *SR* is defined as the percentage of successful runs out of total runs

$$SR = \frac{NSR}{TR} \tag{21}$$

where *NSR* denotes the number of successful runs. A successful run means that all known peaks have been found, that is, $HFP = HKP$. The *AveFEs* over multiple runs can be calculated as

$$AveFEs = \frac{\sum_{i=1}^{TR} FE_i}{TR} \tag{22}$$

where FE_i denotes the number of FEs used to find all peaks in the *i*th run.

In addition, we implement LBPADE using C++ language, and all experiments are executed on a PC with 4 Intel Core i5-3470 3.20-GHz CPUs, 4-GB memory, and a Ubuntu 12.04 LTS 64-bit system.

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

For all test functions, each algorithm terminates if all known peaks have been found or the termination condition is met. The above three metrics–*PR*, *SR*, and *AveFEs*–are used to describe their results.

The results of F1–F20 are shown in Table II with $\varepsilon = 1.0\text{E-}04$. The **boldface** represents the best results in all of the compared algorithms. Meanwhile, the Wilcoxon's rank-sum test [36] ($\alpha = 0.05$) with respect to *PR* between LBPADE and other algorithms is used to examine the significance of the difference.

Table II compares the results of *PR* and *SR* to different multimodal algorithms. The results show that LBPADE performs significantly better than the other algorithms on most test functions. For example, LBPADE obtained the best *PR* results on 15 out of all 20 test functions, among all 11 compared algorithms, as indicated by the **boldface**. It should be noticed that if the algorithm has the same *PR*, the *AveFEs* can be used to measure the performance of the algorithm. Therefore, we compare the *AveFEs* results of LBPADE with CDE, Self_CCDE, NCDE, LIPS, PNPCDE, and MOMMOP on F1–F5. These algorithms are chosen because they are well-performing algorithms in their corresponding categories according to the results in Table II, that is, CDE and Self-CCDE in the niching strategies category, NCDE, LIPS, and PNPCDE in the new evolutionary operators category, and MOMMOP in the multiobjective techniques category. Moreover, F1–F5 are tested because it is not necessary to measure the other complicated functions by the *AveFEs* since the results on *SR* of these algorithms are almost 0 when the algorithms terminate. Herein, we test the *AveFEs* with three different ε levels (i.e., 1.0E-02, 1.0E-03, and 1.0E-04). The detailed comparison results are shown in Table III, and the best results are marked as **boldface**.

1) For the First Five Simple and Not Scalable Functions F1–F5: From the observation of Table II, it can be seen that LBPADE has the best results on F1–F5, and the results are the same as CDE, Self_CCDE, NCDE, PNPCDE, and MOMMOP. Moreover, these algorithms can find all global optima in each run (i.e., *SR* is 1.000). Since LBPADE can form a stable niche based on LBP, each individual is guided by the information of the current niche, which helps LBPADE to deal with MMOPs. Meanwhile, LBPADE performs significantly better than SDE, R2PSO, R3PSO, NSDE, LIPS, and LoICDE on F1–F5.

From Table III, we can find that LBPADE converges faster than all compared algorithms on F1–F5 when $\varepsilon = 1.0\text{E-}02$. Meanwhile, LBPADE obtains the best *AveFEs* values on F1, F2, and F4 when $\varepsilon = 1.0\text{E-}03$, and on F1, F2, and F5 when $\varepsilon = 1.0\text{E-}04$. The significant tests also show that LBPADE generally outperforms all compared algorithms with significantly small *AveFEs* values. These indicate the fast convergence ability of LBPADE.

Therefore, we can conclude that LBPADE generally performs better than the other state-of-the-art multimodal algorithms, on the performance metrics of *PR*, *SR*, and *AveFEs*.

2) For the Scalable Multimodal Functions F6–F10: F6–F10 are the MMOPs that have many global optima, so that many algorithms cannot find all global optima. However, LBPADE has generally better performance on these difficult problems than most of the compared algorithms. Table II shows that only LBPADE, CDE, LoICDE, and MOMMOP can find all global optima on F6 (i.e., *PR* and *SR* are 1.000). For F7, LBPADE obtains the *PR* value of 0.889 and performs better than all others except for MOMMOP. For F8 and F9, LBPADE performs better than many others like CDE, SDE, R2PSO, R3PSO, NCDE, NSDE, LIPS, PNPCDE, and

TABLE II
PR AND *SR* (TEST FUNCTIONS F1–F20)

Func \ Algorithm	LBPADE PR	LBPADE SR	CDE PR	CDE SR	SDE PR	SDE SR	R2PSO PR	R2PSO SR	R3PSO PR	R3PSO SR	Self_CCDE PR	Self_CCDE SR
F1	**1.000**	1.000	**1.000**(≈)	1.000	0.657(+)	0.373	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F2	**1.000**	1.000	**1.000**(≈)	1.000	0.737(+)	0.529	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F3	**1.000**	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F4	**1.000**	1.000	**1.000**(≈)	1.000	0.284(+)	0.000	0.946(+)	0.784	0.966(+)	0.863	**1.000**(≈)	1.000
F5	**1.000**	1.000	**1.000**(≈)	1.000	0.922(+)	0.843	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F6	**1.000**	1.000	**1.000**(≈)	1.000	0.056(+)	0.000	0.537(+)	0.000	0.687(+)	0.000	0.942(+)	0.490
F7	0.889	0.000	0.861(+)	0.000	0.053(+)	0.000	0.484(+)	0.000	0.434(+)	0.000	0.884(+)	0.020
F8	0.575	0.000	0.000(+)	0.000	0.013(+)	0.000	0.023(+)	0.000	0.421(+)	0.000	0.994(-)	0.882
F9	0.476	0.000	0.474(+)	0.000	0.013(+)	0.000	0.122(+)	0.000	0.127(+)	0.000	0.459(+)	0.000
F10	**1.000**	1.000	**1.000**(≈)	1.000	0.147(+)	0.000	0.905(+)	0.353	0.850(+)	0.157	**1.000**(≈)	1.000
F11	0.674	0.000	0.330(+)	0.000	0.314(+)	0.000	0.641(+)	0.000	0.650(+)	0.000	0.778(-)	0.137
F12	0.750	0.000	0.002(+)	0.000	0.208(+)	0.000	0.392(+)	0.000	0.537(+)	0.000	0.422(+)	0.000
F13	0.667	0.000	0.140(+)	0.000	0.297(+)	0.000	0.627(+)	0.000	0.647(+)	0.000	0.653(+)	0.000
F14	**0.667**	0.000	0.024(+)	0.000	0.216(+)	0.000	0.403(+)	0.000	0.637(+)	0.000	0.520(+)	0.000
F15	**0.654**	0.000	0.005(+)	0.000	0.108(+)	0.000	0.103(+)	0.000	0.213(+)	0.000	0.343(+)	0.000
F16	**0.667**	0.000	0.000(+)	0.000	0.108(+)	0.000	0.095(+)	0.000	0.431(+)	0.000	0.655(+)	0.000
F17	**0.532**	0.000	0.000(+)	0.000	0.076(+)	0.000	0.015(+)	0.000	0.096(+)	0.000	0.246(+)	0.000
F18	**0.667**	0.000	0.167(+)	0.000	0.026(+)	0.000	0.036(+)	0.000	0.100(+)	0.000	0.337(+)	0.000
F19	**0.475**	0.000	0.000(+)	0.000	0.105(+)	0.000	0.000(+)	0.000	0.032(+)	0.000	0.113(+)	0.000
F20	**0.275**	0.000	0.000(+)	0.000	0.000(+)	0.000	0.000(+)	0.000	0.078(+)	0.000	0.024(+)	0.000
+ (LBPADE is better)			13		19		16		16		12	
- (LBPADE is worse)			0		0		0		0		2	
≈			7		1		4		4		6	

Func \ Algorithm	NCDE PR	NCDE SR	NSDE PR	NSDE SR	LIPS PR	LIPS SR	PNPCDE PR	PNPCDE SR	LoICDE PR	LoICDE SR	MOMMOP PR	MOMMOP SR
F1	**1.000**(≈)	1.000	**1.000**(≈)	1.000	0.833(+)	0.686	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F2	**1.000**(≈)	1.000	0.776(+)	0.667	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F3	**1.000**(≈)	1.000	**1.000**(≈)	1.000	0.961(+)	0.961	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F4	**1.000**(≈)	1.000	0.240(+)	0.000	0.990(+)	0.961	**1.000**(≈)	1.000	0.975(+)	0.902	**1.000**(≈)	1.000
F5	**1.000**(≈)	1.000	0.745(+)	0.490	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F6	0.305(+)	0.000	0.056(+)	0.000	0.246(+)	0.000	0.537(+)	0.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F7	0.873(+)	0.000	0.053(+)	0.000	0.400(+)	0.000	0.874(+)	0.000	0.705(+)	0.020	**1.000**(-)	1.000
F8	0.002(+)	0.000	0.013(+)	0.000	0.086(+)	0.000	0.000(+)	0.000	0.000(+)	0.000	**1.000**(-)	1.000
F9	0.461(+)	0.000	0.006(+)	0.000	0.108(+)	0.000	0.474(+)	0.000	0.187(+)	0.000	**1.000**(-)	1.000
F10	0.988(+)	0.863	0.098(+)	0.000	0.748(+)	0.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000	**1.000**(≈)	1.000
F11	0.727(-)	0.059	0.248(+)	0.000	**0.974(-)**	0.843	0.667(≈)	0.000	0.660(+)	0.000	0.716(-)	0.020
F12	0.253(+)	0.000	0.135(+)	0.000	0.574(+)	0.000	0.002(+)	0.000	0.495(+)	0.000	**0.939(-)**	0.549
F13	0.667(≈)	0.000	0.225(+)	0.000	**0.794(-)**	0.176	0.461(+)	0.000	0.510(+)	0.000	0.667(≈)	0.000
F14	**0.667**(≈)	0.000	0.190(+)	0.000	0.644(+)	0.000	0.258(+)	0.000	0.657(+)	0.000	**0.667**(≈)	0.000
F15	0.319(+)	0.000	0.125(+)	0.000	0.336(+)	0.000	0.015(+)	0.000	0.299(+)	0.000	0.618(+)	0.000
F16	**0.667**(≈)	0.000	0.170(+)	0.000	0.307(+)	0.000	0.000(+)	0.000	0.556(+)	0.000	0.650(+)	0.000
F17	0.250(+)	0.000	0.108(+)	0.000	0.168(+)	0.000	0.000(+)	0.000	0.222(+)	0.000	0.505(+)	0.000
F18	0.500(+)	0.000	0.163(+)	0.000	0.098(+)	0.000	0.150(+)	0.000	0.219(+)	0.000	0.497(+)	0.000
F19	0.348(+)	0.000	0.098(+)	0.000	0.000(+)	0.000	0.000(+)	0.000	0.032(+)	0.000	0.223(+)	0.000
F20	0.250(+)	0.000	0.123(+)	0.000	0.000(+)	0.000	0.000(+)	0.000	0.126(+)	0.000	0.125(+)	0.000
+	11		18		16		13		14		6	
-	1		0		2		0		0		5	
≈	8		2		2		7		6		9	

'+', '-', and '≈' indicate that the results of the algorithm are significantly better than, worse than, and similar to the ones of LBPADE by Wilcoxon's rank sum test with α=0.05.

LoICDE. Significantly, LBPADE also obtains the best *PR* and *SR* values 1.000 on F10. All of these indicate that LBPADE has a good ability to globally search and can find all or most of the global optima in these complex functions.

3) For the Composition Functions F11–F20: F11–F15 are the composite functions. Moreover, they have many local optima. F16–F20 are also complicated functions. Moreover, F18–F20 are the high-dimensional functions, which are 10*D*, 10*D*, and 20*D*, respectively. It is difficult to measure the performance of each algorithm with *SR*, because the *SR* values of F11–F20 are equal to 0. But the *PR* values of them can reflect the difference of the algorithm performances. Therefore, only the *PR* values of F11–F20 are analyzed in the following contents.

For F11, LBPADE performs better than CDE, SDE, R3PSO, NSDE, and LoICDE. LBPADE performs the best with F12 except for MOMMOP. For F13 and F14, the result of *PR* on LBPADE is the same with NCDE and MOMMOP, while is better than CDE, SDE, R2PSO, R3PSO, Self_CCDE, NSDE, PNPCDE, and LoICDE. For F15, it is worth noticing that only LBPADE and MOMMOP can obtain *PR* results larger than 0.5 (i.e., find more than half of the global optima). Moreover, LBPADE is also the winner, whose *PR* value is 0.654, which is significantly better than that of MOMMOP.

When dealing with F16–F20, most of the algorithms show poor performance, and some algorithms even cannot deal with these test functions. LBPADE still performs better than CDE, SDE, R2PSO, R3PSO, Self_CCDE, NCDE, NSDE, LIPS, PNPCDE, LoICDE, and MOMMOP on F16–F20. Especially, for F18 to F20, LBPADE obtains the best results among all compared algorithms. It indicates that LBPADE has a strong ability to globally search when dealing with the high-dimensional functions.

TABLE III
AveFEs OF DIFFERENT ALGORITHMS

			ε=1.0E-02				
Func	LBPADE	CDE	Self_CCDE	NCDE	LIPS	PNPCDE	MOMMOP
F1	**1.20E+02**	1.60E+02 (+)	3.55E+02 (+)	6.85E+02 (+)	1.64E+04 (+)	1.62E+02 (+)	1.20E+02 (≈)
F2	**1.52E+02**	4.36E+02 (+)	4.02E+02 (+)	4.58E+02 (+)	3.04E+02 (+)	3.97E+02 (+)	6.40E+02 (+)
F3	**6.50E+01**	3.64E+02 (+)	2.23E+02 (+)	2.21E+02 (+)	2.60E+02 (+)	3.37E+02 (+)	3.38E+02 (+)
F4	**3.07E+03**	1.61E+04 (+)	4.15E+03 (+)	3.60E+03 (+)	3.49E+03 (+)	1.21E+04 (+)	4.52E+03 (+)
F5	**3.70E+02**	1.52E+03 (+)	9.68E+03 (+)	7.84E+02 (+)	5.07E+02 (+)	1.33E+03 (+)	5.81E+02 (+)
+ (LBPADE is better)		5	5	5	5	5	4
- (LBPADE is worse)		0	0	0	0	0	0
≈		0	0	0	0	0	1
			ε=1.0E-03				
Func	LBPADE	CDE	Self_CCDE	NCDE	LIPS	PNPCDE	MOMMOP
F1	**1.60E+02**	**1.60E+02 (≈)**	3.55E+02 (+)	6.85E+02 (+)	1.64E+04 (+)	**1.62E+02 (≈)**	3.32E+02 (+)
F2	**4.45E+02**	1.39E+03 (+)	7.61E+02 (+)	9.27E+02 (+)	4.93E+02 (+)	1.08E+03 (+)	2.51E+03 (+)
F3	5.55E+02	1.11E+03 (+)	5.55E+02 (≈)	4.94E+02 (-)	1.40E+03 (+)	8.33E+02 (+)	**2.04E+02 (-)**
F4	**3.26E+03**	2.71E+04 (+)	7.07E+03 (+)	3.73E+03 (+)	4.28E+03 (+)	2.28E+04 (+)	2.26E+04 (+)
F5	1.38E+03	5.42E+03 (+)	1.97E+03 (+)	**1.57E+02 (-)**	9.35E+02 (-)	3.69E+04 (+)	1.38E+03 (≈)
+ (LBPADE is better)		4	4	3	4	4	3
- (LBPADE is worse)		0	0	2	1	0	1
≈		1	1	0	0	1	1
			ε=1.0E-04				
Func	LBPADE	CDE	Self_CCDE	NCDE	LIPS	PNPCDE	MOMMOP
F1	**1.60E+02**	**1.60E+02 (≈)**	7.18E+02 (+)	6.88E+02 (+)	1.66E+04 (+)	1.65E+02 (+)	5.23E+02 (+)
F2	1.71E+03	3.23E+03 (+)	2.36E+04 (+)	1.66E+03 (-)	**7.62E+02 (-)**	2.66E+03 (+)	3.62E+03 (+)
F3	1.10E+03	3.36E+03 (+)	1.25E+03 (+)	**9.84E+02 (-)**	2.68E+03 (+)	2.43E+03 (+)	2.85E+03 (+)
F4	**3.62E+03**	3.93E+04 (+)	4.42E+04 (+)	4.96E+03 (+)	5.00E+03 (+)	3.38E+04 (+)	5.63E+04 (+)
F5	**1.43E+03**	1.12E+04 (+)	1.28E+04 (+)	2.48E+03 (+)	1.50E+03 (+)	8.99E+03 (+)	**1.43E+03 (≈)**
+ (LBPADE is better)		4	5	3	4	5	4
- (LBPADE is worse)		0	0	2	1	0	0
≈		1	0	0	0	0	1

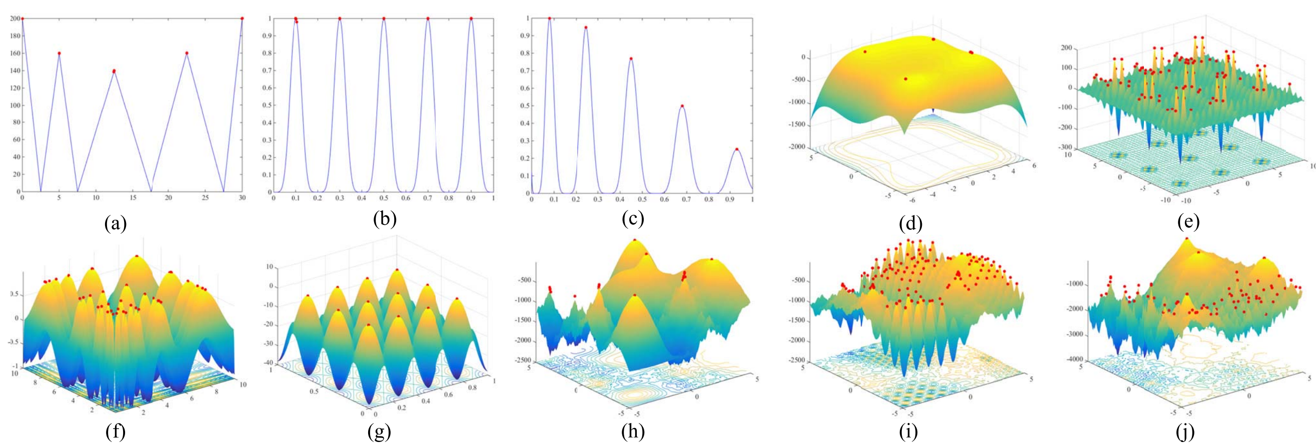

Fig. 3. Final solutions distribution on ten selected functions: (a) F1. (b) F2. (c) F3. (d) F4. (e) F6. (f) F7. (g) F10. (h) F11. (i) F12. (j) F13.

In general, LBPADE performs better than other algorithms on most test functions. We can also conclude that CDE, SDE, R2PSO, R3PSO, NSDE, and LIPS perform better on simple and low-dimensional functions, but their abilities of dealing with complicated problems are poor. Self_CCDE, PNPCDE, LoICDE, and MOMMOP also perform poorly when dealing with high-dimensional functions. In contrast, LBPADE always performs better in both low- and high-dimensional functions. Overall, the LBPADE is feasible and promising in solving most of the tested MMOPs.

To further investigate the LBPADE, the final solution distributions of some functions (i.e., F1, F2, F3, F4, F6, F7, F10, F11, F12, and F13) are shown in Fig. 3, which is formed when the algorithm achieved the *MaxFEs* of test functions or all of the global optima have been found in the evolutionary process.

From Fig. 3, we find that LBPADE can find most of the global optima on these functions that have only a few numbers of global optima, such as Fig. 3(a)–(d). More important, for the functions with a large number of global optima, LBPADE also can find all of the global optima, such as Fig. 3(e)–(g). Moreover, for some complicated functions, LBPADE still can find most of the global optima, such as Fig. 3(h)–(j). Therefore, LBPADE has a good ability to locate global optima in MMOPs.

Moreover, we compared the runtime of each algorithm on F1–F20 at accuracy 1.0E-04. The results are presented in Table S.III in the supplementary material. From Table S.III

TABLE IV
EXPERIMENTAL RESULT IN *PR* AND *SR* BETWEEN LBPADE AND THEIR VARIANTS WITH DIFFERENT MUTATION STRATEGIES ON F1–F20 WITH ACCURACY LEVEL $\varepsilon = 1.0\text{E-}04$

Variant / Func	LBPADE		LBPADE-best		LBPADE-rand		LBPADE-pbest(A)		LBPADE-pbest		LBPADE-crand	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F2	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F3	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F4	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	0.955(+)	0.818
F5	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F6	**1.000**	1.000	0.707(+)	0.000	0.966(+)	0.941	0.979(+)	0.636	0.808(+)	0.364	0.763(+)	0.000
F7	**0.889**	0.000	0.475(+)	0.000	0.836(+)	0.000	0.836(+)	0.000	0.838(+)	0.000	0.823(+)	0.000
F8	0.575	0.000	0.361(+)	0.000	**0.845**(−)	0.000	0.482(+)	0.000	0.040(+)	0.000	0.027(+)	0.000
F9	**0.476**	0.000	0.176(+)	0.000	0.378(+)	0.000	0.413(+)	0.000	0.420(+)	0.000	0.424(+)	0.000
F10	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F11	**0.674**	0.000	0.667(+)	0.000	0.606(+)	0.000	0.667(+)	0.000	0.591(+)	0.000	0.576(+)	0.000
F12	**0.750**	0.000	0.625(+)	0.000	0.090(+)	0.000	0.739(+)	0.000	0.057(+)	0.000	0.068(+)	0.000
F13	**0.667**	0.000	0.632(+)	0.000	0.394(+)	0.000	**0.667(≈)**	0.000	0.439(+)	0.000	0.470(+)	0.000
F14	**0.667**	0.000	0.603(+)	0.000	0.500(+)	0.000	**0.667(≈)**	0.000	**0.667(≈)**	0.000	0.530(+)	0.000
F15	**0.654**	0.000	0.570(+)	0.000	0.159(+)	0.000	0.568(+)	0.000	0.182(+)	0.000	0.205(+)	0.000
F16	**0.667**	0.000	**0.667(≈)**	0.000	0.287(+)	0.000	**0.677(≈)**	0.000	**0.667(≈)**	0.000	0.409(+)	0.000
F17	**0.532**	0.000	0.409(+)	0.000	0.102(+)	0.000	0.441(+)	0.000	0.148(+)	0.000	0.136(+)	0.000
F18	**0.667**	0.000	0.620(+)	0.000	0.167(+)	0.000	**0.667(≈)**	0.000	0.475(+)	0.000	0.167(+)	0.000
F19	**0.475**	0.000	0.400(+)	0.000	0.000(+)	0.000	0.361(+)	0.000	0.167(+)	0.000	0.011(+)	0.000
F20	**0.275**	0.000	0.200(+)	0.000	0.000(+)	0.000	0.236(+)	0.000	0.125(+)	0.000	0.023(+)	0.000
+ (LBPADE is better)			13		13		10		12		15	
- (LBPADE is worse)			0		1		0		0		0	
≈			7		6		10		8		5	

in the supplementary material, we find that the runtime of LBPADE is promising compared with the other algorithms. Generally speaking, although LBPADE spends slightly more time than some algorithms like CDE, SDE, R2PSO, R3PSO, and Self_CCDE, the runtime differences are acceptable when combined with the comparisons on solution quality. More important, LBPADE spends less time than other algorithms like NCDE, NSDE, LIPS, PNPCDE, LoICDE, and MOMMOP, almost on all of the functions. Therefore, the proposed LBPADE has a very promising runtime in handling MMOPs.

C. Effects of the NGI Mutation Strategy

In this section, we investigate the effect of the NGI mutation strategy by comparing it with the results derived from the usage of different mutation strategies on LBPADE, such as DE/rand/1 and DE/best/1, which are denoted as LBPADE-rand and LBPADE-best, respectively. Meanwhile, we also compare the LBPADE with better DE mutation strategy, such as DE/current-to-*p*best/1 with or without archive and DE/current-to-rand/1, which are denoted as LBPADE-pbest(A), LBPADE-pbest, and LBPADE-crand, respectively. The *PR* and *SR* results of different LBPADE variants on $\varepsilon = 1.0\text{E-}04$ are listed in Table IV.

From Table IV, we find that LBPADE significantly outperforms all compared mutation strategies on most test functions. For F1 to F5, all LBPADE variants have promising results. LBPADE locates all of the global optima in each run with F6, which performs best among all LBPADE variants. For F7, all LBPADE variants obtain a promising result except LBPADE-best. This may be because the DE/best/1mutation strategy is somehow greedy to make some individuals trapped into local optima. But for F8, LBPADE obtains slightly worse *PR* values than LBPADE-rand due to F8 having many unevenly distributed global optima. It should be noted that LBPADE performs better than other LBPADE variants as the function dimension increases. This phenomenon can be seen from the *PR* values on F11–F20. That is, LBPADE obtains higher *PR* results than LBPADE-best, LBPADE-rand, LBPADE-pbest(A), LBPADE-pbest, and LBPADE-crand on these functions. It indicates that LBPADE has better global searchability when dealing with complicated or high-dimensional functions. In conclusion, our NGI mutation strategy is a promising method by combining the niching information from the LBP niche and the global information from the entire population, which helps LBPADE to solve the MMOPs more efficiently.

D. Advantage of APS

To validate the advantage of APS, five different variants of LBPADE are designed as follows. The first variant, namely, LBPADE-CR, fixes F and lets CR be controlled by APS. The second variant, namely, LBPADE-F lets F be controlled by APS but fixes CR. The third variant, namely, LBPDE, fixes F and CR at 0.5 and 0.1, respectively. Besides, some other adaptive parameter mechanisms are also used to compare with our APS, such as the adapt mechanisms introduced by JADE [19] and LSHADE [37], which are denoted as LBP-JADE and LBP-LSHADE, respectively. Table V shows the *PR* and *SR* results of different LBPADE variants.

In Table V, **boldface** denotes the best *PR* values among LBPADE and its five variants. For F1 to F6, the *PR* values are 1.000 across all algorithms except for LBP-JADE, which might cause the adaptive parameter mechanisms introduced

TABLE V
EXPERIMENTAL RESULT IN *PR* AND *SR* RESULT IN LBPADE WITH DIFFERENT PARAMETER SETTINGS ON F1–F20 WITH ACCURACY LEVEL ε = 1.0E-04

Variant / Func	LBPADE		LBPADE-CR		LBPADE-F		LBPDE		LBP-JADE		LBP-LSHADE	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F2	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F3	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F4	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	0.773(+)	0.000	**1.000(≈)**	1.000
F5	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000
F6	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	1.000	0.000(+)	0.000	**1.000(≈)**	1.000
F7	0.889	0.000	0.745(+)	0.000	**0.891(-)**	0.157	0.752(+)	0.000	0.720(+)	0.000	0.856(+)	0.078
F8	0.575	0.000	**0.898(-)**	0.097	0.612(-)	0.000	0.696(-)	0.000	0.455(+)	0.000	0.073(+)	0.000
F9	**0.476**	0.000	0.389(+)	0.000	0.416(+)	0.000	0.375(+)	0.000	0.247(+)	0.000	0.455(+)	0.000
F10	**1.000**	1.000	**1.000(≈)**	1.000	**1.000(≈)**	**1.000**	**1.000(≈)**	1.000	0.765(+)	0.000	**1.000(≈)**	1.000
F11	**0.674**	0.000	0.667(+)	0.000	0.667(+)	0.000	0.667(+)	0.000	0.606(+)	0.000	0.667(+)	0.000
F12	**0.750**	0.000	0.284(+)	0.000	0.738(+)	0.000	0.329(+)	0.000	0.068(+)	0.000	0.000(+)	0.000
F13	**0.667**	0.000	**0.667(≈)**	0.000	**0.667(≈)**	0.000	**0.667(≈)**	0.000	0.485(+)	0.000	**0.667(≈)**	0.000
F14	**0.667**	0.000	0.650(+)	0.000	**0.667(≈)**	0.000	**0.667(≈)**	0.000	0.152(+)	0.000	**0.667(≈)**	0.000
F15	**0.654**	0.000	0.375(+)	0.000	0.500(+)	0.000	0.375(+)	0.000	0.000(+)	0.000	0.375(+)	0.000
F16	**0.667**	0.000	**0.667(≈)**	0.000	**0.667(≈)**	0.000	0.635(+)	0.000	**0.667(≈)**	0.000	**0.667(≈)**	0.000
F17	**0.532**	0.000	0.409(+)	0.000	0.420(+)	0.000	0.340(+)	0.000	0.000(+)	0.000	0.341(+)	0.000
F18	**0.667**	0.000	0.621(+)	0.000	**0.667(≈)**	0.000	0.604(+)	0.000	0.561(+)	0.000	0.606(+)	0.000
F19	**0.475**	0.000	0.400(+)	0.000	0.400(+)	0.000	0.325(+)	0.000	0.000(+)	0.000	0.170(+)	0.000
F20	**0.275**	0.000	0.200(+)	0.000	0.193(+)	0.000	0.136(+)	0.000	0.159(+)	0.000	0.216(+)	0.000
+ (LBPADE is better)			10		7		10		15		10	
- (LBPADE is worse)			1		2		1		0		0	
≈			9		11		9		5		10	

by JADE to trap some individuals into local optima when dealing with F4 and F6. For F7, LBPADE, LBPADE-F, and LBP-LSHADE achieve nearly the same results larger than 0.8, which are 0.889, 0.891, and 0.856, respectively. For F8, LBPADE performs slightly worse than LBPADE-CR and LBPDE, but it still has competitive performance compared to LBPADE-F and LBP-LSHADE. For F9, LBPADE significantly outperforms the other mutation variants except for LBP-LSHADE. We can also find that the *PR* values of LBPADE and LBP-LSHADE on F9 are 0.476 and 0.475, respectively, which indicate that the adaptive parameter mechanisms of LBPADE and LBP-LSHADE are effective to F9. Besides, LBPADE, LBPADE-CR, LBPADE-F, LBPDE, and LBP-LSHADE perform best on F10, F13, F14, and F16, whose *PR* values are 1.000, 0.667, 0.667, and 0.667, respectively. With respect to the remaining test function (i.e., F12, F15, and F17–F20), LBPADE still performs best in all of the variants, and such results indicate that the APS has an increasing positive effect as the function dimension rises. It is thus confirmed that the APS is helpful for LBPADE when dealing with not only simple functions but also high-dimensional and complicated functions.

Besides, for a more comprehensive analysis of LBPADE, we further investigated the contributions of the two components (i.e., APS and NGI) in LBPADE. In fact, the contribution of APS can be obtained from the LBPADE variant without NGI, namely, the LBPADE-rand, the LBPADE-best, and the other LBPADE variant in Table IV. Similarly, the contribution of NGI can be obtained from the LBPADE variant without APS, such as LBPADE-F, LBP-JADE, and others in Table V.

To avoid the potential bias caused by selecting inappropriate counterparts, we adopt a novel way herein to estimate the contributions made by each component. First, according to Table IV, we calculate the value of "number of $+$" minus "number of $-$" for each compared variant. For example, for the LBPADE-best, the value is $13-0=13$, and the value for LBPADE-crand is $15-0=15$. Then, the average of these values is calculated and the result is $(13+12+10+12+15)/5=12.4$. We can regard that this value can reflect the contribution that NGI brings to LBPADE, the larger the value, the more contributions it brings. Similarly, we can obtain this value from Table V as $(9+5+9+15+10)/5=9.6$ to indicate the contribution that APS brings to LBPADE. From the above analyses and the result $(12.4>9.6)$, we can conclude that NGI generally contributes more than APS to the performance of LBPADE.

E. Maintaining the Identified Optima

During evolution, the MMOPs require that the algorithm can locate many global optima simultaneously. The algorithm with good performance not only can maintain the global optima that have been identified (found) but also can continue to search for the other global optima that have not been found. To investigate the performance of LBPADE on maintaining the identified optima, the solution distribution of some functions (i.e., F2, F4, and F10) is presented on some specific generations. Fig. 4 shows the solution distribution of F2 with different generations (i.e., the generations are 1, 10, 20, and the final generation, respectively). We can find that LBPADE

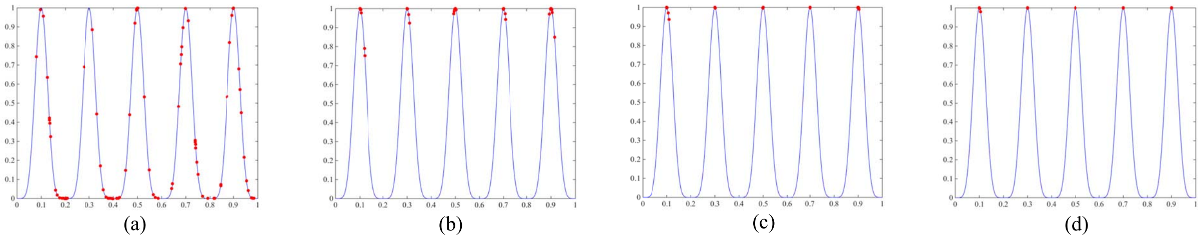

Fig. 4. Distribution of solutions with evolutions on F2. (a) Generation = 1. (b) Generation = 10. (c) Generation = 20. (d) Final generation.

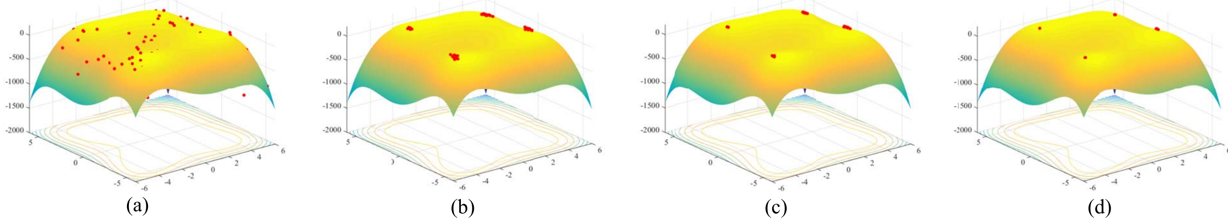

Fig. 5. Distribution of solutions with evolutions on F4. (a) Generation = 1. (b) Generation = 30. (c) Generation = 50. (d) Final generation.

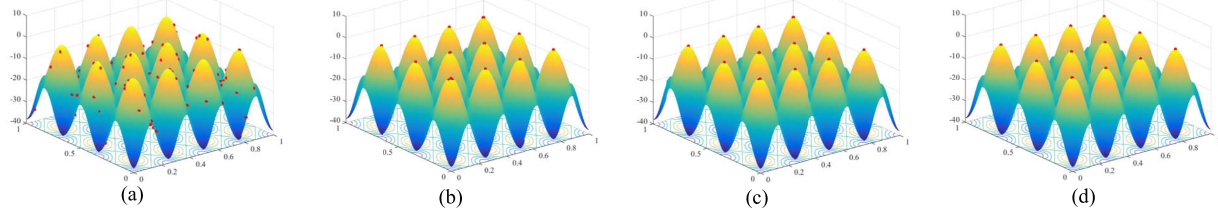

Fig. 6. Distribution of solutions with evolutions on F10. (a) Generation = 1. (b) Generation = 50. (c) Generation = 100. (d) Final generation.

has obtained all of the global optima when the generation is 20 on F2 [i.e., Fig. 4(c)]. It indicates that LBPADE can locate the global optima quickly, namely, LBPADE has a good ability of global search. Fig. 5 shows the solution distribution of F4 with different generations (i.e., 1, 30, 50, and the final generation, respectively). From Fig. 5, we can find that the population nearly converges completely when the generation is 30, but some solutions still do not achieve the convergence state until the generation is 50. Similarly, the solution distributions of F10 with different generations are presented in Fig. 6. With the increasing generation, all solutions gradually achieve the convergence state. All global optima are found until the generation is 100.

Overall, it is clear that the found solutions are not dispersed with the increasing evolution in LBPADE. Namely, our LBPADE can maintain the global optima until the end of evolution.

F. Impacts of Parameter Settings

In this section, we mainly investigate the impacts of parameters λ and μ on our LBPADE. The best *PR* are highlighted in **boldface**, and the last row of the table called "#Best" counts the number of the best *PR* each algorithm obtains on the total 20 functions, namely, the number of the bolded *PR*.

1) Effect of Parameter λ: The parameter λ is used in (12) to control the state of exploration and exploitation during the evolution process. Table S.IV in the supplementary material shows the *PR* results with different values of λ.

From Table S.IV in the supplementary material, we have two observations. It is found that a small λ value will cause most of the *FEs* to be exhausted during exploitation. This may lead to the ineffectiveness in exploring the search space, and it may degrade the diversity of solutions. If the λ value is large, most *FEs* are exhausted during exploration, which may lead to a poor ability of global search. Therefore, selecting a suitable λ value is important to balance the state of exploitation and exploration during evolution. We find that the #Best is 20 when λ is 0.8 in Table S.IV in the supplementary material, which means that LBPADE performs best with different λ. For F1 to F5, there is not much difference as the value of λ varies. The reason is that F1 to F5 have achieved convergence in the early stage, so the value of λ has less of an effect on F1 to F5. For F6 to F20, LBPADE also performs the best when λ is 0.8. It indicates that $\lambda = 0.8$ is a reasonable setting.

2) Effect of Parameter μ: As introduced in Section III-E, μ is a parameter of BCS, which controls the way to deal with boundary during the evolution process. To investigate the effect of μ on LBPADE's performance, two groups of possible μ values are tested. In group 1, μ is set as 0, 1E-05, 1E-04, 1E-03, and 1E-02, respectively. In group 2, μ is set as 0.1, 0.2, 0.4, 0.5, 0.6, 0.8, and 1, respectively. It should be noticed that with μ being equal to 1, the range-exceeded solution is reset to the boundary (L_j or U_j). With μ being equal to 0, the range-exceeded solution is reset to the best solution ($\boldsymbol{x}_{n\text{best}}$) in the current niching. The result of *PR* under different settings of μ is shown in Table S.V in the supplementary material.

From Table S.V in the supplementary material, it is clear that the μ has a significant effect on high-dimensional functions (i.e., F11–F20). For F1, the *PR* result is only 0.900 when μ is 0, implying that by directly assigning range-exceeded solutions to $\boldsymbol{x}_{n\text{best}}$, the population diversity may decrease and thus degrade the algorithmic performance. As μ increases, the *PR* results are all equal to 1.000 on F2–F6 and F10. However, for the high-dimensional functions, such as F7, F9, F12, F15, F17, and F20, the best values of *PR* are achieved when μ is 1E-04. It indicates that a suitable value of μ is helpful for the LBPADE to locate more global optima. Moreover, the effect of μ on LBPADE becomes more and more obvious as the function dimension increases.

G. Comparisons With Winners of CEC Competitions

In the above experiments, LBPADE shows general better performance than the compared state-of-the-art multimodal algorithms when dealing with MMOPs. To further prove the effective performance of LBPADE, we compare LBPADE with the winners of the CEC'2013 and CEC'2015 competitions on multimodal optimization, which are nearest-better clustering (NEA2) [38] and the niching migratory multiswarm optimizer (NMMSO) [39], respectively. Meanwhile, we also compare LBPADE with a dynamic archive-niching DE algorithm (dADE2) [40]. For simplicity and convenience, we directly cite the results of these algorithms from the corresponding competitions, where the results of NEA2 and dADE2 are from https://github.com/mikeagn/CEC2013/tree/master/NichingCompetition2013FinalData and the results of NMMSO are from https://github.com/mikeagn/CEC2013/tree/master/NichingCompetition2015FinalData.

Tables S.VI–S.X in the supplementary material present the comparison results with respect to *PR* and *SR* between LBPADE and these algorithms (NEA2, NMMSO, and dADE2). The results of accuracy level from 1.0E-01 to 1.0E-05 are shown in Tables S.VI–S.X in the supplementary material, respectively. The best *PR* are highlighted in **boldface**, and the last row of the table called #Best counts the number of the best *PR* each algorithm obtains on the total 20 functions, namely, the number of bolded *PR*.

From Tables S.VI–S.X in the supplementary material, we can draw the following conclusions.

1) LBPADE performs the best in all compared algorithms at the accuracy levels $\varepsilon = 1.0\text{E-}01$ and 1.0E-02. More specifically, at $\varepsilon = 1.0\text{E-}01$, LBPADE can find all of the peaks in each run except for F9, F12, and F19. Particularly, at these accuracy levels, LBPADE is much better than NEA2, NMMSO, and dADE2 on F15–F20 that have many local optima.
2) At accuracy level $\varepsilon = 1.0\text{E-}03$, LBPADE and NMMSO generally perform better than NEA2 and dADE2. They both perform best in 12 of all test problems. But the difference is that LBPADE has great advantages in dealing with high-dimensional problems, while NMMSO performs better in low-dimensional problems. For example, LBPADE performs better than NMMSO on F15–F20 which are in high dimension.
3) At the last two accuracy levels, LBPADE performs better than dADE2, it also remains at its competitive performance with NEA2 and NMMSO. Even though the number of #Best for LBPADE, NEA2, and NMMSO is 8, 12, and 12 at $\varepsilon = 1.0\text{E-}04$ and 8, 10, and 11 at $\varepsilon = 1.0\text{E-}05$, respectively, LBPADE can achieve very similar performance to NEA2 and NMMSO on most of those functions. For example, with respect to *PR* at $\varepsilon = 1.0\text{E-}04$, on F16 and F18, LBPADE achieves 0.667 and 0.667, respectively, which is very similar to NEA2 with 0.673 and 0.667, respectively.

Overall, we can see that LBPADE is competitive against the winners of the CEC'2013 and the CEC'2015 competitions.

V. Conclusion

In this paper, a novel LBP-based niching strategy is proposed, which forms a niche for each individual according to its local information by simulating the LBP operator in the image processing. Meanwhile, the NGI mutation strategy and the APS technique are incorporated, which results in the LBPADE algorithm, which not only can enhance the population diversity but also can accelerate convergence speed. The experimental results show that the proposed LBPADE can outperform a number of state-of-the-art multimodal optimization algorithms on benchmark problems.

In the future, we will extend the LBPADE to solve some real-world problems in the domains with potential multimodal optimization requirements, including resource-constrained project scheduling [41]; electricity markets [42]; energy resource management [43]; optical networks [44]; cloud computing [45], [46]; and image processing [47], [48]. For example, how to detect multiple equilibriums simultaneously is a key challenging economic game problem in electricity markets. Herein, we can use the following two main advantages of LBPADE to solve this problem. One is that the NGI mutation strategy of LBPADE can locate multiple optima simultaneously and avoid solutions trapped into local optima; and the other is that the APS of LBPADE does not need to input the extra sensitive parameters when dealing with this problem. So LBPADE has a promising ability to solve these real-world problems. Besides, we will try to use the parallel/distributed computing resources to reduce the runtime of LBPADE and further improve the performance of LBPADE.

References

[1] R. Storn and K. Price, "Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, Feb. 1995.

[2] 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.

[3] Y.-L. Li, Z.-H. Zhan, Y.-J. Gong, J. Zhang, Y. Li, and Q. Li, "Fast micro-differential evolution for topological active net optimization," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1411–1423, Jun. 2016.

[4] 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.

[5] Y.-L. Li, Z.-H. Zhan, Y.-J. Gong, W.-N. Chen, J. Zhang, and Y. Li, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798–1810, Sep. 2015.

[6] 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.

[7] E. C. Osuna and D. Sudholt, "Runtime analysis of crowding mechanisms for multimodal optimisation," *IEEE Trans. Evol. Comput.*, to be published. doi: 10.1109/TEVC.2019.2914606.

[8] Y. H. Li, Z.-H. Zhan, S. J. Lin, J. Zhang, and X. N. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems," *Inf. Sci.*, vol. 293, no. 1, pp. 370–382, Feb. 2015.

[9] 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.

[10] Q. Yang *et al.*, "Adaptive multimodal continuous ant colony optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 191–205, Apr. 2017.

[11] Q. Yang, W.-N. Chen, Y. Li, C. L. P. Chen, X.-M. Hu, and J. Zhang, "Multimodal estimation of distribution algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 636–650, Mar. 2017.

[12] Y. L. Cao, H. Zhang, W. F. Li, M. C. Zhou, Y. Zhang, and W. A. Chaovalitwongse, "Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions," *IEEE Trans. Evol. Comput.*, to be published. doi: 10.1109/TEVC.2018.2885075.

[13] 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.

[14] 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.

[15] F. Caraffini and A. V. Kononova, "Structural bias in differential evolution: A preliminary study," in *Proc. AIP Conf.*, vol. 2070, no. 1, 2019, Art. no. 020005.

[16] F. Caraffini, A. V. Kononova, and D. Corne, "Infeasibility and structural bias in differential evolution," *Inf. Sci.*, vol. 496, pp. 161–179, Sep. 2019.

[17] A. Yaman, G. Iacca, and F. Caraffini, "A comparison of three differential evolution strategies in terms of early convergence with different population sizes," in *Proc. AIP Conf.*, vol. 2070, no. 1, 2019, Art. no. 020002.

[18] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[19] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

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

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

[22] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 150–169, Feb. 2010.

[23] 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.

[24] I. Poikolainen, F. Neri, and F. Caraffini, "Cluster-based population initialization for differential evolution frameworks," *Inf. Sci.*, vol. 297, pp. 216–235, Mar. 2015.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] Z.-H. Zhan, J. J. Li, J. N. Cao, J. Zhang, H. S.-H. Chung, and Y. H. Shi, "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[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.*, to be published. doi: 10.1109/TEVC.2018.2875430.

[31] 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.

[32] R. Cheng, M. Q. 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.

[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] 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.

[35] X. D. 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, Tech. Rep., 2013. Accessed: Aug. 2019. [Online]. Available: http://titan.csit.rmit.edu.au/∼e46507/cec13-niching/competition/cec2013-niching-benchmark-tech-report.pdf

[36] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[37] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 1658–1665.

[38] M. Preuss, "Niching the CMA-ES via nearest-better clustering," in *Proc. Genet. Evol. Comput. Conf.*, Portland, OR, USA, 2010, pp. 1711–1718.

[39] J. E. Fieldsend, "Running up those hills: Multi-modal search with the niching migratory multi-swarm optimiser," in *Proc. IEEE. Congr. Evol. Comput.*, Beijing, China, 2014, pp. 2593–2600.

[40] M. G. Epitropakis, X. Li, and E. K. Burke, "A dynamic archive niching differential evolution algorithm for multimodal optimization," in *Proc. Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 79–86.

[41] S. Elsayed, R. Sarker, T. Ray, and C. Coello Coello, "Consolidated optimization algorithm for resource-constrained project scheduling problems," *Inf. Sci.*, vols. 418–419, pp. 346–362, Dec. 2017.

[42] 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.

[43] F. Lezama, L. E. Sucar, E. M. De Cote, J. Soares, and Z. Vale, "Differential evolution strategies for large-scale energy resource management in smart grids," in *Proc. Genet. Evol. Comput. Conf.*, Berlin, Germany, 2017, pp. 1279–1286.

[44] F. Callegati, L. H. Bonani, F. Lezama, W. Cerroni, A. Campi, and G. Castanon, "Trunk reservation for fair utilization in flexible optical networks," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 889–892, May 2014.

[45] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. L. 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.

[46] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Comput. Surveys*, vol. 47, no. 4, pp. 1–33, Jul. 2015.

[47] J. Yu, X. Yang, F. Gao, and D. Tao, "Deep multimodal distance metric learning using click constraints for image ranking," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4014–4024, Dec. 2017.

[48] D. Y. Tian, D. Y. Zhou, M. G. Gong, and Y. W. Wei, "Interval type-2 fuzzy logic for semisupervised multimodal hashing," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2906658.

Hong Zhao (S'17) received the Bachelor's degree in computer science from Henan Polytechnic University, Jiaozuo, China, in 2014, and the Master's degree in computer science from Kunming University of Science and Technology, Kunming, China, in 2017. She is currently pursuing the Ph.D. degree in computer science with the South China University of Technology, Guangzhou, China.

Her current research interests include artificial intelligence, evolutionary computation, swarm intelligence, and their applications in design and optimization.

Zhi-Hui Zhan (M'13–SM'18) received the bachelor's and Ph.D. degrees in computer science from 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 artificial intelligence, evolutionary computation, swarm intelligence, and their applications in real-world problems, and in environments of cloud computing and big data.

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

Ying Lin (M'12) received the Ph.D. degree in computer applied technology from Sun Yat-sen University, Guangzhou, China, in 2012.

She is currently an Associate Professor with the Department of Psychology, Sun Yat-sen University. Her current research interests include computational intelligence and its applications in network analysis and cognitive diagnosis.

Xiaofeng Chen received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 1998 and 2000, respectively, and the Ph.D. degree in cryptography from Xidian University, Xi'an, in 2003.

He is currently a Professor with Xidian University. He has published over 200 research papers in refereed international conferences and journals. His work has been cited over 7000 times at Google Scholar. His current research interests include applied cryptography and cloud computing security.

Prof. Chen is on the editorial board of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, *Security and Privacy*, and *Computing and Informatics*. He has served as the program/general chair or a program committee member in over 30 international conferences.

Xiao-Nan Luo received the B.S. degree in computational mathematics from Jiangxi University, Nanchang, China, the M.S. degree in applied mathematics from Xidian University, Xi'an, China, and the Ph.D. degree in computational mathematics from the Dalian University of Technology, Dalian, China.

He is currently a Professor with the School of Computer and Information Security, Guilin University of Electronic Technology, Guilin, China. He received the National Science Fund for Distinguished Young Scholars granted by the National Natural Science Foundation of China, and was the Director of the National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou, China. His current research interests include computer graphics, machine learning, and pattern recognition.

Jie Zhang received the master's degree in computer science from the China University of Mining and Technology, Xuzhou, China, in 1999.

She is currently an Associate Professor with the Beijing University of Chemical Technology, Beijing, China. Her current research interests include formal verification, and PHM for power and embedded system design.

Sam Kwong (F'13) received the B.Sc. degree in electrical engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in 1983, the M.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree in electrical engineering from the University of Hagen, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Ottawa, ON, Canada, where he designed the diagnostic software to detect the manufactured faults of the very large-scale integration chips in the Cyber 430 machine. He later joined Bell Northern Research Canada, Ottawa, as a Scientific Staff Member. In 1990, he joined the City University of Hong Kong, Hong Kong, as a Lecturer with the Department of Electronic Engineering, where he is currently a Professor with the Department of Computer Science. His current research interests include pattern recognition, evolutionary computations, and video analytics.

Prof. Kwong was elevated to an IEEE Fellow for his contributions on optimization techniques for cybernetics and video coding in 2014. He was also appointed IEEE Distinguished Lecturer of the IEEE Systems, Man and Cybernetics (SMC) Society in 2017. He is currently the Vice President of IEEE SMC on Cybernetics.

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.