A Cluster-Based Differential Evolution With Self-Adaptive Strategy for Multimodal Optimization

Weifeng Gao, Gary G. Yen, *Fellow, IEEE,* and Sanyang Liu

***Abstract*—Multimodal optimization is one of the most challenging tasks for optimization. It requires an algorithm to effectively locate multiple global and local optima, not just single optimum as in a single objective global optimization problem. To address this objective, this paper first investigates a cluster-based differential evolution (DE) for multimodal optimization problems. The clustering partition is used to divide the whole population into subpopulations so that different subpopulations can locate different optima. Furthermore, the self-adaptive parameter control is employed to enhance the search ability of DE. In this paper, the proposed multipopulation strategy and the self-adaptive parameter control technique are applied to two versions of DE, crowding DE (CDE) and species-based DE (SDE), which yield self-CCDE and self-CSDE, respectively. The new algorithms are tested on two different sets of benchmark functions and are compared with several state-of-the-art designs. The experiment results demonstrate the effectiveness and efficiency of the proposed multipopulation strategy and the self-adaptive parameter control technique. The proposed algorithms consistently rank top among all the competing state-of-the-art algorithms.**

***Index Terms*—Clustering method, differential evolution, multipopulation, niching, self-adaptive strategy.**

I. Introduction

UNLIKE the single objective optimization problems (SOPs) that have been widely studied in the literature, most practical optimization problems need to simultaneously locate multiple global and local optima of a given objective function. Pattern matching and recognition is a typical example. Given a template pattern, all patterns that match the template perfectly, or not so perfectly, are of interest. A perfect match corresponds to a global optimum, whereas not so perfect matches correspond to local optima. These local optima should not be dismissed, as they represent target patterns, though imperfectly formed. For real world problems due to physical (and/or cost) constraints, the best results cannot be realized at times. Under such conditions, if multiple solutions (local and global) are known, the implementation can be quickly switched to an alternative solution while still maintaining an acceptable system performance. Multiple solutions could also be analyzed to discover hidden properties (or relationships) of the concerned functional landscape. These problems are referred to as multimodal optimization problems. As the name suggests, this requires optimization algorithms to find multiple optimal solutions (both local and global) and not just one single optimum as is done in a typical optimization study. If a classical derivative-based optimization method is used for multimodal optimization, the approach must have to be applied several times, every time with the expectation of finding a different optimal solution.

In recent years, as an important branch of derivative-free methods, evolutionary algorithms (EAs), which are population-based approaches, have attracted a growing interest. However, the original EAs are usually designed for locating a single global solution. These algorithms typically converge to one final solution because of the global selection scheme used. To solve multimodal optimization problems, numerous techniques have been developed [1]–[5].

In this paper, we focus on differential evolution (DE), proposed by Storn and Price [6]. DE perturbs the current-generation population members with the scaled differences of randomly selected and distinct population members. Therefore, no separate probability distribution has to be used, which makes the scheme self-organizing in this respect. It is a new branch of EAs, and has shown a very promising searching ability for SOPs [7]–[13]. However, like other EAs, it is difficult for the basic DE to optimize multimodal functions. The difficulty lies in how to locate multiple global and local optima and how to maintain the identified optima until the end of the search. In the basic DE, it is a thorny issue to deal with the above two problems due to diversity loss. The diversity loss is mainly due to the mutation operation and the greedy selection scheme, which result in that all the individuals quickly converge on a local or global optimum, where the current best individual locates. This feature is beneficial for SOPs. However, for multimodal optimization problems, this feature goes against DE to locate the multiple global and local optima. For multimodal optimization problems, it is important to guide individuals to move toward different promising subregions. Nevertheless, the question is how to assign individuals to move toward different subregions and how to drive individual toward an optimum as soon as possible.

Several DE algorithms have been recently proposed to address multimodal optimization problems using the multipopulation method [14]– [18], which seems an ideal

Manuscript received April 12, 2013; revised July 17, 2013; accepted September 11, 2013. Date of publication October 4, 2013; date of current version July 15, 2014. This work was supported by the National Nature Science Foundation of China under Grant 61373174. This paper was recommended by Associate Editor S. Mostaghim.

W. Gao and S. Liu are with the School of Science, Xidian University, Xi'an 710071, China (e-mail: gaoweifeng2004@126.com; liusanyang@126.com).

G. G. Yen is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA (e-mail: gyen@okstate.edu).

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.2013.2282491

technique. The multipopulation method can be used to enhance the diversity of the population, with the aim of maintaining multiple subpopulations on different peaks. The traditional methods of using the multipopulation to find optima divide the whole search space into local subspaces; each of which might cover one or a small number of local optima, and then separately searches within these subspaces. However, the difficulty is how to define the area for each subregion in the search space and how to generate subpopulations.

In recent years, adaptive or self-adaptive parameter control, has attracted considerable attentions. If properly designed, adaptive strategy can enhance the robustness of an algorithm by dynamically adapting the parameters to the characteristics of different fitness landscapes. The convergence rate can be improved if the control parameters are adapted to appropriate values at different evolution stages of a specific problem. Several kinds of adaptive or self-adaptive strategy have been developed [19]–[21]. The experiment results show that these strategies can significantly improve the performance of the basic DE. However, these methods are mainly centered on SOPs. There are still many problems remaining to be answered for multimodal optimization problems, e.g., how to design a self-adaptive strategy that is well suited for multimodal optimization problems?

Motivated by the above two main issues, this paper proposes a cluster-based differential evolution with self-adaptive strategy for multimodal optimization. The proposed algorithm has the following two features.

1) The proposed approach adopts multipopulation method for multimodal optimization problems to improve the search ability. The whole population is divided into subpopulations by the clustering method. The clustering partition can enable the algorithm to assign individuals to different promising subregions.
2) The proposed approach adopts a self-adaptive parameter control technique in DE. Due to the control parameters adapted to appropriate values based on different evolution stages, it can improve not only the convergence rate but also the accuracy.

The clustering partition is incorporated into two common niching DE algorithms, i.e., crowding DE (CDE) [22] and species-based DE (SDE) [23], which yield CCDE and CSDE. Then, the self-adaptive strategy based CCDE (self-CCDE) and self-adaptive strategy based CSDE (self-CSDE) are presented. The computational results demonstrate the effectiveness of the self-adaptive strategy and the multipopulation method.

The remainder of this paper is organized as follows. Section II discusses existing literatures in DE. In Section III, the proposed algorithm is presented in sufficient details. The problem definition and experiment results are presented and discussed in Sections IV and V, respectively. Finally, the paper is concluded in Section VI with pertinent observations.

II. Basic DE and Related Studies

A. Differential Evolution

The DE, which was proposed by Storn and Price [6], implements mutation, crossover, and selection operations to update the population during the evolution. There are several versions of DE ever designed. The one used in this paper is called DE/rand/1.

The population of DE consists of Np n-dimensional real-valued vectors

$$X_i = \{x_{i,1}, x_{i,2}, ..., x_{i,n}\}, \qquad i = 1, 2, ..., Np. \tag{1}$$

Taking into account of each individual X_i (called a target vector), a mutant vector $V_i = \{v_{i,1}, v_{i,2}, ..., v_{i,n}\}$ is obtained by adding the weighted difference of two population members to a third individual

$$V_i = X_{r_1} + F \cdot (X_{r_2} - X_{r_3}) \tag{2}$$

where r_1, r_2, and r_3 are randomly selected from $\{1, 2, ..., Np\}$ and satisfy $r_1 \neq r_2 \neq r_3 \neq i$, and F is the scaling factor.

After the mutation operation, the trial vector $U_i = \{u_{i,1}, u_{i,2}, ..., u_{i,n}\}$ is generated by making use of a binomial crossover operation on the target vector X_i and the mutant vector V_i

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } (rand_j(0,1) \leq Cr) \text{ or } (j = j_{rand}) \\ x_{i,j} & \text{otherwise} \end{cases} \tag{3}$$

where $i = 1, 2, ..., Np$, $j = 1, 2, ..., n$, j_{rand} is a randomly chosen integer from $\{1, 2, ..., n\}$, $rand_j(0, 1)$ is a uniformly distributed random number between 0 and 1, which is generated for each j, and $Cr \in [0, 1]$ is called the crossover probability. Due to the use of j_{rand}, the trial vector U_i is guaranteed to differ from its target vector X_i.

The selection operation is conducted by comparing the target vector X_i against the trial vector U_i. The better one will enter the next generation. For the maximization problem

$$X_i = \begin{cases} U_i & \text{if } f(U_i) > f(X_i) \\ X_i & \text{otherwise.} \end{cases} \tag{4}$$

B. DE For Multimodal Optimization Problems

Many researchers have attempted to extend DE to deal with multimodal optimization problems and several multimodal optimization DE algorithms have been proposed in the literatures [14]–[18], [22], [23], and [28]–[34]. In the following, a brief survey of literature on DE for multimodal optimization can be mainly classified as two categories.

1) Multipopulation Methods: Many experimental studies have shown that the multipopulation method is an effective approach to enhancing the diversity for DE to solve SOPs and multimodal optimization problems. Clustering-based approaches are one of the most effective implementations of multipopulation methods.

Halder *et al.* [24] presented a cluster-based dynamic DE with external archive for global optimization in dynamic fitness landscape. The algorithm uses a multipopulation method where the entire population is partitioned into several clusters according to the spatial locations of the trial solutions. Cai *et al.* [25] proposed a new method, called learning-enhanced DE for SOPs, which adopts a clustering-based learning strategy. Cai *et al.* [26] developed a clustering-based DE. In this method, the one-step k-means clustering acts as several

multiparent crossover operators to utilize the information of the population efficiently, and hence, it can enhance the performance of DE. Wang *et al.* [27] designed a dynamic clustering based DE for SOPs. Simulation results show that this method improves solution accuracy with less computational effort.

The first attempt of using DE to evolve multiple subpopulations for solving multimodal optimization problems dates back to 1999 and was due to Rigling and Moore [14]. The proposed algorithm implements a mating restriction, so that the mutation operations are performed within each subpopulation. Additionally, a penalty is applied to the members of each subpopulation that are too close to the members of different subpopulations to drive each subpopulation toward a different optimum. The method requires definition of the number of subpopulations, a penalty term, and the minimum spanning distance, which are all problem-dependent parameters and thus form a major difficulty for the use of the method. Rumbler and Moore [15] attempted to overcome these limitations by suggesting a new method for determining the optimal values for these parameters. The idea is to simply run the algorithm repeatedly with different parametric setups to determine suitable values and at the same time keep record of the found solutions. As expected the repeated runs increase the computational complexity of the whole process.

Zaharie [16] proposed a multiresolution multipopulation DE (MMDE). The main idea of MMDE is to divide the population into equally sized subpopulations. In each subpopulation, a general DE is applied for a given number of generations. After this, a migration process is implemented to maintain the diversity and avoid premature convergence. When a subpopulation converged, the found optima are collected into an archive. However, it is possible that some optima are missed during a single search epoch if there are multiple optima close to each other. To solve this problem, the repeated search epochs are used and each searching epoch is characterized by a specific resolution factor. MMDE does not require the definition of the niche radius, but introduces a set of new parameters: the number and size of the subpopulations, as well as the number and length of the epochs. Hendershot [17] presented an extension of DE (named MultiDE), which also adopts subpopulations strategy. However, in MultiDE the number of subpopulations is kept variable, i.e., subpopulations can emerge and disappear. MultiDE uses a structure similar to archive in MMDE, called population 0. When an element from a subpopulation is similar to an element from population 0, the former is no longer considered for further evolutions. The similarity is based on a precision parameter to be controlled by the user. MultiDE employs a minimum spanning distance to encourage the search for different optima. It also introduces another time delay regarding the tunable parameter operation, i.e., the number of generations after which a subpopulation is eliminated if it fails to discover a new optimum. At the same year, Zaharie [18] developed a hybrid DE algorithm which combines a crowding technique with the multipopulation method. This method has two characteristics.

1) Since the crowing technique allows each subpopulation to locate many optima the subpopulation reinitialization is no longer necessary.
2) The crowding computation is limited to subpopulations, thus a global processing step can be avoided.

Algorithm 1 Crowding DE (CDE)

Step 1 Randomly generate Np number of initial trial solutions.

Step 2 For $i = 1$ to Np Produce an offspring U_i using the standard DE. Calculate the Euclidean distances of U_i to the other individuals in the DE population Compare the fitness of U_i with the most similar individual and replace it if the U_i has a better fitness value.
End For

Step 3 Stop if a termination criterion is satisfied. Otherwise go to Step 2.

2) Niching Technique: While the study is not limited to the multipopulation method and more work about niching technique can be seen as follows.

Thomsen [22] integrated the fitness sharing concept with DE to form the Sharing DE. Sharing DE utilizes the classical sharing technique, based on the Euclidean distance. In each generation, the number of offspring is equal to the parent population size. Thus, after Np trial vectors have been generated from Np parents, the sharing function is used to calculate the fitness for each individual and the worst half of the population is eliminated. However, the algorithm requires defining a niche radius. In order to address the issue, Thomsen [22] proposed to extend DE with a crowding scheme (CDE) to allow it to tackle multimodal optimization problems, as presented in Algorithm 1. In CDE, when an offspring is generated by using the standard DE, it competes only with the most similar individual (measured by Euclidean distance) in the current population. The offspring will replace this individual if it has a better fitness value. To avoid replacement error in CDE, the crowding factor is chosen to be equal to the population size Np.

Species-based DE (SDE) [23] is another commonly used DE niching algorithm, as presented in Algorithm 2. The concept is the same as speciation described in [1]. Different niches are formed around the species seed. The mutation is carried out within each species. The pseudocodes of CDE and SDE are presented in Algorithms 1 and 2.

Recently, DE with local selection (DELS) where the target vector and the base vector for mutation are kept the same, has been demonstrated to outperform the classic DE with global selection (DEGS) in convex unimodal problems [28]. In order to exploit the potential of DELS for solving multimodal problems, Rönkkönen and Lampinen [28] divided the mutation operation of DELS into two parts: local mutation and global mutation. A selective probability is used to control the frequency of adopting local mutation and global mutation. Wong *et al.* [29] incorporated the principles of spatial and temporal locality into CDE. Qu *et al.* [30] used a neighborhood mutation strategy that is performed within each Euclidean neighborhood to improve the performance of DE. Roy *et al.* [31] proposed a two-stage hybrid multimodal optimizer based on invasive weed optimization (IWO) and DE for locating and preserving

Algorithm 2 Species-based DE (SDE)

Step 1 Randomly generate *Np* number of initial trial solutions.

Step 2 Sort all individuals in descending order of their fitness values.

Step 3 Determine the species seeds for the current population: The most-fit individual will be set as the first species seed. Then all individuals are checked in turn from the most-fit to the least-fit against the species seeds found so far. If an individual does not fall in the radius of any seeds, it will be identified as another species seed.

Step 4 For each species, execute the basic DE:

4.1 If a species has less than *m* individuals, randomly generate new individuals within the radius of the species seed.

4.2 If the offspring's fitness is the same as its species seed, replace this offspring with a randomly generated new individual.

Step 5 Keep only the *Np* fitter individuals from the combined population.

Step 6 Stop if a termination criterion is satisfied. Otherwise go to Step 2.

multiple optima of a real-parameter functional landscape. Most recently inspired by the two multiobjective approaches for solving multimodal optimization problems: the bi-objective multipopulation genetic algorithm [32] and the niching-based nondominated sorting genetic algorithm (NSGA-II) of Deb and Saha [33], Basak *et al.* [34] proposed a simple but effective bi-objective formulation of the multimodal optimization problems. Simulation results suggest that this method is able to provide better and more consistent performance over the existing well known multimodal algorithms.

C. Adaptive DE for Single Objective Optimization Problems

Generally, the optimal parameter and operator settings of DE are problem dependent. The simplest method for parameter and operator settings is by trial and error, but this is inconvenient in practice; therefore, adaptive control techniques have been utilized recently. For example, Brest *et al.* [19] proposed a self-adaptive DE (jDE), where the values of F and Cr are either inherited from parents or randomly generated numbers. During the evolution, the new F takes a value from 0.1 to 0.9 in a random manner with a probability τ_1, the new Cr takes a value from 0.0 to 1.0 in a random manner with a probability τ_2, and both of them are obtained before the mutation is executed. Qin *et al.* [20] also proposed a self-adaptive DE (SaDE) by controlling the mutation schemes and parameters dynamically based on previous search experience. Zhang and Sanderson [21] proposed an adaptive DE (JADE) using a new current-to-pbest mutation scheme to generate mutation vectors. The values of F and Cr are generated by a Cauchy distribution and a normal distribution, respectively. JADE extracts information from the recent successful F and Cr and uses such information for generating new F and Cr. The experiment results in [21] show that JADE is better than, or at least comparable to, other adaptive DE algorithms, on problem set used.

III. Proposed Algorithm

As mentioned above, the main objective of this paper is to assess the contribution of incorporating multipopulation strategy and the self-adaptive parameter control technique into DE for multimodal optimization problems. The proposed algorithm is designed with this motivation. The two key components of the proposed algorithm are the multipopulation management strategy and the self-adaptive parameter control technique. The two strategies work together to improve the performance of DE as found by the experiment results in Section V. In the following subsection, we will detail the two techniques and explain why the two techniques can work together seamlessly.

A. Multipopulation Strategy

In the basic DE, the mutation operation is performed between randomly picked individuals from the entire population. This will allow any two members to generate the difference vector in DE/rand/1, even if the two members are far apart from each other. This mutation operation is efficient when searching for single global solution. The basic DE also apply the greedy selection scheme. Both designs result in the diversity loss as the evolution process moves on. At last, all individuals in general evolve to one local or global optimal point. However, when solving a multimodal problem, multiple optima need to be located simultaneously. If the global version of DE is used, it will not be efficient for multiple localized convergences at the final search stage as required for multimodal optimization. At the final search stage, the whole population is distributed around different optimal regions. If the distance in the parameter space between different optima is large, efficient convergence to any of the optima will become impossible as the difference vectors can be generated using individuals from different optimal regions with relatively large magnitudes. Hence, how to maintain the diversity of the population so as to provide an opportunity that every optimal region can be covered by some individuals, is a challenging issue.

The multipopulation method is a priming way to enhance the diversity of the population, with the aim of maintaining multiple populations on different optima. However, the traditional multipopulation method [16]–[18] is that the whole search space is divided into multiple subdomains by resolution factors, then each subpopulation is initialized with random individuals selected from a subdomain. The experiment results reported in [16] and [17] show that these methods have difficulty in tracking all the peaks. To address this situation, inspired by the clustering technique successfully used in single objective optimization and dynamic optimization [24]–[27], we introduce a clustering partition to divide the whole population into a number of subpopulations, which cover different local regions, based on spatial positions of the individuals. It can enable the algorithm to assign individuals to different promis-

Algorithm 3 The clustering partition

Step 1 Randomly produce a reference point R from the search space.

Step 2 Find the nearest individual $\tilde{X} \in P$ to R, which means that compared with the other individuals in P, $\tilde{X}$ has the minimum Euclidean dista- nce from R. If more than one individual has the minimum Euclidean distance, $\tilde{X}$ is chosen randomly.

Step 3 Combine $M-1$ individuals of the population P, which are nearest to $\tilde{X}$, with $\tilde{X}$ to form a subpopulation. Any tie will be broken randomly.

Step 4 Eliminate these M individuals from P.

Step 5 Execute Step 2-Step 4 repeatedly until the population P is divided into Np/M subpopulations.

Algorithm 4 Self-CCDE

Step 1 Set $Crm = 0.5$ and initialize a population P of Np individuals $\{X_i | i = 1, 2, ..., Np\}$ in the search region randomly.

Step 2 Divide population into subpopulations ($subpop_j$, $j = 1, 2, ..., M$) by the clustering partition.

For each subpopulation $subpop_j$

For each individual in $subpop_j$

2.1 Pick $r1, r2, r3$ from the $subpop_j$ and generate mutation vector using Eq. (5).

2.2 Generate the crossover probability and use the crossover operation of DE to produce the trial vector.

End For

End For

Step 3 Set $S_{Cr} = \emptyset$. For $i = 1$ to Np

3.1 Evaluate offspring U_i using the fitness function.

3.2 Compare the fitness of U_i with the most similar individual X_s (in Euclidean distance) in population. If $f(U_i) \geq f(X_s)$, $X_s = U_i$, save the respective crossover probability in S_{Cr}

End For

Step 4 Update $Crm = mean(S_{Cr})$.

Step 5 Stop if a termination criterion is satisfied. Otherwise go to Step 2.

ing subregions and automatically calculate the search region for each subpopulation.

It works as follows. First, the population of size Np is split into a group of disjoint subpopulations according to the location of individuals in the solution space, each of which comprises M individuals with adjacent location. Then, we use M individuals of each subpopulation to generate the same number of offspring by DE, which is likely to move the subpopulation toward near peak. The population P with size Np is clustered into a number of subpopulations with size M as presented in Algorithm 3.

Note that the above clustering partition is not strictly the classical clustering method, such as k-means method or hierarchical clustering method. In this paper, we focus on dividing the whole population into a number of subpopulations.

B. Self-Adaptive Strategy

Based on the analysis of the last subsection, we can maintain the diversity of the population by the multipopulation method. Another problem appears how to make the algorithm quickly converge to the optima contained by different subpopulations. From the literatures, many algorithms have been proposed to address this problem using the adaptive or self-adaptive parameter control method [19]–[21], which seems an ideal technique to improve the search ability of DE. On this recommendation, we develop a self-adaptive DE. First, a mutation operation is given as follows:

$$V_i = X_{r_1} + \frac{f(X_{r_2}) - f(X_{r_3})}{fbest - fworst} \cdot (X_{r_2} - X_{r_3}) \tag{5}$$

where r_1, r_2, and r_3 are randomly selected from $\{1, 2, ..., Np\}$ and satisfy $r_1 \neq r_2 \neq r_3 \neq i$, $fbest$ is the best fitness of the individual in the current population and $fworst$ is the worst fitness. For maximization problem, the larger the final result, the better it is. It is known that the second term in (5) is the guidance term, which controls the search direction. The new mutation operation is expected to offer an opportunity that the search moves toward promising direction. It is because if $f(X_{r_2}) > f(X_{r_3})$, $(X_{r_2} - X_{r_3})$ is in the promising direction to exploit, and if $f(X_{r_2}) < f(X_{r_3})$, the actually search direction is $(X_{r_3} - X_{r_2})$, which is the promising direction to explore. What is more, the new mutation operation realizes a self-adaptive parameter control strategy on F.

Furthermore, we adopt a simple self-adaptive strategy to dynamically update Cr. This is similar to the Cr adaptation strategy used in JADE [21].

At each generation, for each target vector, the crossover rate Cr_i is independently generated according to a normal distribution of mean Crm and standard deviation 0.1

$$Cr_i = randn(Crm, 0.1) \tag{6}$$

and truncated to the interval [0, 1], where Crm is the mean value to generate Cr_i. It is updated as follows:

$$Crm = mean(S_{Cr}) \tag{7}$$

where $mean()$ is the usual arithmetic mean operation; and S_{Cr} is the set of all successful crossover rates Cr_i at previous generations.

C. Algorithmic Framework

It is important to emphasize that in this paper, we intend to make use of the multipopulation strategy and the self-adaptive parameter control technique to improve the performance of CDE and SDE, which are two popular DE algorithms to deal with multimodal optimization problems. The pseudocode of the cluster-based CDE with self-adaptive strategy (self-CCDE), and the cluster-based SDE with self-adaptive strategy (self-CSDE) are presented in Algorithms 4 and 5, respectively. Note that the original SDE adopts the greedy selection scheme. In order to maintain the diversity of the population in self-CSDE, it also employs the same selection scheme as CDE.

TABLE I
TEST FUNCTIONS FOR EXPERIMENT ONE (CF: COMPOSITION FUNCTION)

Test Function Set 1		Test Function Set 2	
Test Function name / Dimensions	Number of Global/Local peaks	Test Function name / Dimensions	Number of Global peaks
F_1: two-peak trap/1D	1/1	CF_1: /10D and 30D	8
F_2: central two-peak trap/1D	1/1	CF_2: /10D and 30D	6
F_3: five-uneven-peak trap/1D	2/3	CF_3: /10D and 30D	6
F_4: equal maxima/1D	5/0	CF_4: /10D and 30D	6
F_5: decreasing maxima/1D	1/4	CF_5: /10D and 30D	6
F_6: uneven maxima/1D	5/0	CF_6: /10D and 30D	6
F_7: uneven decreasing maxima/1D	1/4	CF_7: /10D and 30D	6
F_8: Himmelblaus function/2D	4/0	CF_8: /10D and 30D	6
F_9: six-hump camel back/2D	2/2	CF_9: /10D and 30D	6
F_{10}: Shekels foxholes/2D	1/24	CF_{10}: /10D and 30D	6
F_{11}: 2-D inverted Shubert function/2D	18/many	CF_{11}: /10D and 30D	8
F_{12}: 1-D inverted Vincent function/1D	6/0	CF_{12}: /10D and 30D	8
F_{13}: 2-D inverted Vincent function/2D	36/0	CF_{13}: /10D and 30D	10
F_{14}: 3-D inverted Vincent function/3D	216/0	CF_{14}: /10D and 30D	10
		CF_{15}: /10D and 30D	10

TABLE II
TEST FUNCTION SETTING

Function No.	ε	r	Population size	No. of function evaluations
F_1	0.05	0.5	50	10,000
F_2	0.05	0.5	50	10,000
F_3	0.05	0.5	50	10,000
F_4	0.000001	0.01	50	10,000
F_5	0.000001	0.01	50	10,000
F_6	0.000001	0.01	50	10,000
F_7	0.000001	0.01	50	10,000
F_8	0.0005	0.5	50	10,000
F_9	0.000001	0.5	50	10,000
F_{10}	0.00001	0.5	50	10,000
F_{11}	0.05	0.5	250	100,000
F_{12}	0.0001	0.2	100	20,000
F_{13}	0.001	0.2	500	200,000
F_{14}	0.001	0.2	1000	400,000
$CF_1 - CF_{15}$ in D=10	0.5	1	600	300,000
$CF_1 - CF_{15}$ in D=30	1	1	1000	800,000

Taking Self-CCDE as an example, we analyze the complexity of the algorithm. Compared to CDE, the additional operators added in Self-CCDE are the clustering partition and the self-adaptive strategy. The computational complexity of the clustering partition is $O(D \cdot Np^2)$ where D is the dimension of decision space and Np is population size. The complexity of the self-adaptive strategy is $O(Np)$. Since the complexities of the crowding scheme and original DE are $O(D \cdot Np^2)$ and $O(D \cdot Np)$, respectively, the complexity of CDE is $O(D \cdot Np^2)$. As a result, the computational complexity of self-CCDE remains to be $O(D \cdot Np^2)$.

Algorithm 5 Self-CSDE

Step 1 Set $Crm = 0.5$ and initialize a population P of Np individuals $\{X_i | i = 1, 2, ..., Np\}$ in the search region randomly.

Step 2 Sort all individuals in descending order of their fitness values.

Step 3 For $j = 1$ to Np/M Determine the species seed $\bar{X}$ which is the best (fitness value) unprocessed individual. Combine $M - 1$ individuals of the population P, which are nearest to $\bar{X}$, with $\bar{X}$ to form the subpopulation $subpop_j$. Eliminate these M individuals from the current populations.
End For

Step 4 Set $S_{Cr} = \emptyset$.
For each subpopulation $subpop_j$
For each individual in $subpop_j$

2.1 Pick $r1, r2, r3$ from the $subpop_j$ and generate mutation vector using Eq. (5).

2.2 Generate the crossover probability and use the crossover operation of DE to produce the trial vector.

2.3 Evaluate offspring U_i using the fitness function.

2.4 Compare the fitness of U_i with the most similar indi- vidual X_s (in Euclidean distance) in $subpop_j$.

If $f(U_i) \geq f(X_s)$, $X_s = U_i$, save the respective crossover probability in S_{Cr}
End For
End For

Step 5 Update $Crm = mean(S_{Cr})$.

Step 6 Stop if a termination criterion is satisfied. Otherwise go to Step 2.

IV. EXPERIMENTAL SETTING

A. Benchmark Problems

To evaluate the performance of the proposed algorithms, we use a set of 14 general multimodal functions (i.e., $F_1 - F_{14}$) [34], [36] and a set of 15 scalable composition functions (i.e., $CF_1 - CF_{15}$) [37]. The functions are divided into two classes: test function set 1 and test function set 2, as shown in Table I. These functions are widely used in published papers and are very difficult to track the peaks. For example, for $F_1 - F_3$, the existing local optima can misguide the population to move away from the true global optimum; $F_4 - F_7$ have global peaks, while for F_4 the five peaks are evenly spaced, for F_5, F_7 the five peaks decrease in height exponentially, and for F_6 the peaks are unevenly spaced. F_8 has four global peaks with two closer to each other than the other two. Unlike functions from test function set 1, no exact number of local peaks is made available for the fifteen scalable composite functions by the authors in [37].

TABLE III

AVERAGE NUMBER OF GLOBAL PEAKS FOUND FOR TEST FUNCTION SET 1 AND TEST FUNCTION SET 2 WITH D=10 AND THE RESPECTIVE RANKS (IN PARENTHESES)

Fun	Self-CCDE	Self-CSDE	CCDE	CSDE	CDE [22]	SDE [23]	FER-PSO [36]	SPSO [36]	r2pso [36]	r3pso [36]	SCMA [38]	CMA[38]
F_1	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	$^{\dagger}0.72^{\dagger}$ (11)	$^{\dagger}0.48^{\dagger}$ (12)	$^{\dagger}0.76^{\dagger}$ (10)	$^{\dagger}0.84^{\dagger}$ (9)	1 (1)	1 (1)
F_2	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	$^{\dagger}0.44^{\dagger}$ (12)	$^{\dagger}0.88^{\dagger}$ (11)	$^{\dagger}0.96^{\dagger}$ (10)	1 (1)	1 (1)
F_3	2 (1)	2 (1)	2 (1)	2 (1)	2 (1)	$^{\dagger}1.96^{\dagger}$ (6)	$^{\dagger}0.8^{\dagger}$ (9)	$^{\dagger}0.24^{\dagger}$ (12)	$^{\dagger}0.48^{\dagger}$ (11)	$^{\dagger}0.6^{\dagger}$ (10)	$^{\dagger}1.92^{\dagger}$ (8)	$^{\dagger}1.95^{\dagger}$ (7)
F_4	5 (1)	5 (1)	5 (1)	5 (1)	$^{\dagger}3.84^{\dagger}$ (10)	$^{\dagger}4.72^{\dagger}$ (8)	$^{\dagger}4.84^{\dagger}$ (6)	$^{\dagger}4.88^{\dagger}$ (5)	$^{\dagger}4.68^{\dagger}$ (9)	$^{\dagger}4.74^{\dagger}$ (7)	$^{\dagger}0.04^{\dagger}$ (12)	$^{\dagger}0.6^{\dagger}$ (11)
F_5	1 (1)	1 (1)	1 (1)	1 (1)	$^{\dagger}0.72^{\dagger}$ (12)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
F_6	5 (1)	5 (1)	5 (1)	$^{\dagger}4.8^{\dagger}$ (7)	$^{\dagger}3.96^{\dagger}$ (10)	$^{\dagger}4.6^{\dagger}$ (9)	5 (1)	4.92 (5)	$^{\dagger}4.88^{\dagger}$ (6)	$^{\dagger}4.72^{\dagger}$ (8)	$^{\dagger}0^{\dagger}$ (12)	$^{\dagger}0.64^{\dagger}$ (11)
F_7	1 (1)	1 (1)	1 (1)	1 (1)	$^{\dagger}0.6^{\dagger}$ (12)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
F_8	4 (1)	4 (1)	$^{\dagger}3.56^{\dagger}$ (6)	$^{\dagger}3.8^{\dagger}$ (3)	$^{\dagger}0.32^{\dagger}$ (12)	$^{\dagger}3.72^{\dagger}$ (4)	$^{\dagger}3.68^{\dagger}$ (5)	$^{\dagger}0.84^{\dagger}$ (11)	$^{\dagger}2.92^{\dagger}$ (9)	$^{\dagger}2.76^{\dagger}$ (10)	$^{\dagger}3.44^{\dagger}$ (7)	$^{\dagger}3.43^{\dagger}$ (8)
F_9	2 (1)	2 (1)	$^{\dagger}1.86^{\dagger}$ (7)	2 (1)	$^{\dagger}0.04^{\dagger}$ (12)	2 (1)	1.96 (6)	$^{\dagger}0.08^{\dagger}$ (11)	$^{\dagger}1.44^{\dagger}$ (10)	$^{\dagger}1.56^{\dagger}$ (9)	$^{\dagger}1.6^{\dagger}$ (8)	2 (1)
F_{10}	1 (1)	1 (1)	0.92 (5)	1 (1)	$^{\dagger}0.52^{\dagger}$ (9)	$^{\dagger}0.32^{\dagger}$ (10)	1 (1)	$^{\dagger}0.56^{\dagger}$ (8)	$^{\dagger}0.88^{\dagger}$ (6)	$^{\dagger}0.76^{\dagger}$ (7)	$^{\dagger}0.04^{\dagger}$ (12)	$^{\dagger}0.18^{\dagger}$ (11)
F_{11}	18 (1)	18 (1)	18 (1)	18 (1)	$^{\dagger}17.7^{\dagger}$ (5)	$^{\dagger}12.4^{\dagger}$ (9)	$^{\dagger}17.4^{\dagger}$ (6)	$^{\dagger}8.52^{\dagger}$ (12)	$^{\dagger}15.2^{\dagger}$ (8)	$^{\dagger}15.6^{\dagger}$ (7)	$^{\dagger}12.04^{\dagger}$ (10)	$^{\dagger}12^{\dagger}$ (11)
F_{12}	6 (1)‡	$^{\dagger}5.89$ (3)	$^{\dagger}5.9$ (2)	$^{\dagger}5.8$ (5)	$^{\dagger}5.56^{\dagger}$ (7.5)	$^{\dagger}4.88^{\dagger}$ (12)	$^{\dagger}5.36^{\dagger}$ (10)	$^{\dagger}5.6^{\dagger}$ (6)	$^{\dagger}5.52^{\dagger}$ (9)	$^{\dagger}5.16^{\dagger}$ (11)	$^{\dagger}5.56^{\dagger}$ (7.5)	$^{\dagger}5.81^{\dagger}$ (4)
F_{13}	35.94‡ (1)	$^{\dagger}33$ (4)	$^{\dagger}34.6^{\ddagger}$ (2)	$^{\dagger}30.2^{\dagger}$ (5)	$^{\dagger}33.8^{\ddagger}$ (3)	$^{\dagger}22.8^{\dagger}$ (10)	$^{\dagger}23.6^{\dagger}$ (9)	$^{\dagger}25.7^{\dagger}$ (6)	$^{\dagger}21.8^{\dagger}$ (12)	$^{\dagger}22.2^{\dagger}$ (11)	$^{\dagger}24.6^{\dagger}$ (7)	$^{\dagger}23.7^{\dagger}$ (8)
F_{14}	185.22‡ (1)	$^{\dagger}118.86$ (4)	$^{\dagger}160.12^{\ddagger}$ (2)	$^{\dagger}102.50^{\dagger}$ (5)	$^{\dagger}152^{\ddagger}$ (3)	$^{\dagger}50.6^{\dagger}$ (8)	$^{\dagger}68.6^{\dagger}$ (7)	$^{\dagger}70.1^{\dagger}$ (6)	$^{\dagger}40.6^{\dagger}$ (10)	$^{\dagger}45.4^{\dagger}$ (9)	$^{\dagger}0.6^{\dagger}$ (12)	$^{\dagger}32.5^{\dagger}$ (11)
Total ranks	14	22	32	34	98.5	81	74	108	113	110	99.5	87

TABLE IV

AVERAGE NUMBER OF GLOBAL PEAKS FOUND FOR TEST FUNCTION SET 2 WITH D=10 AND THE RESPECTIVE RANKS (IN PARENTHESES)

Fun	Self-CCDE	Self-CSDE	CCDE	CSDE	CDE	SDE	FER-PSO	SPSO	r2pso	r3pso	SCMA	CMA
CF_1	5.6†(3)	$^{\ddagger}6.9$ (1)	$^{\dagger}5.1^{\dagger}$ (4)	$^{\ddagger}6.6^{\dagger}$ (2)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.79^{\dagger}$ (6)	$^{\dagger}1.08^{\dagger}$ (7)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}2^{\dagger}$ (5)	$^{\dagger}1^{\dagger}$(8)
CF_2	4 (1)	4 (1)	$^{\dagger}3.7^{\dagger}$ (4)	4 (1)	$^{\dagger}1.2^{\dagger}$ (9)	$^{\dagger}1.3^{\dagger}$ (8)	$^{\dagger}2^{\dagger}$ (5)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}1.52^{\dagger}$ (6)	$^{\dagger}1.4^{\dagger}$ (7)
CF_3	6 (1)	6 (1)	6 (1)	6 (1)	$^{\dagger}0.7^{\dagger}$ (9)	$^{\dagger}1.5^{\dagger}$ (8)	$^{\dagger}2.5^{\dagger}$ (5)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}2.21^{\dagger}$ (6)	$^{\dagger}2.08^{\dagger}$ (7)
CF_4	5.2†(3)	$^{\ddagger}5.6$ (1)	$^{\dagger}4.7^{\dagger}$ (4)	$^{\ddagger}5.42^{\dagger}$ (2)	$^{\dagger}0^{\dagger}$ (9.5)	$^{\dagger}0^{\dagger}$ (9.5)	$^{\dagger}0^{\dagger}$ (9.5)	$^{\dagger}0^{\dagger}$ (9.5)	$^{\dagger}0^{\dagger}$ (9.5)	$^{\dagger}0^{\dagger}$ (9.5)	$^{\dagger}0.2^{\dagger}$ (6)	$^{\dagger}0.4^{\dagger}$ (5)
CF_5	5.5†(3)	‡ 6 (1)	$^{\dagger}5.3^{\dagger}$ (4)	$^{\ddagger}6$ (1)	$^{\dagger}1.1^{\dagger}$ (8)	$^{\dagger}1.3^{\dagger}$ (6)	$^{\dagger}2^{\dagger}$ (5)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}0^{\dagger}$ (11)	$^{\dagger}1.2^{\dagger}$ (7)	$^{\dagger}1^{\dagger}$ (9)
CF_6	3 (1)	3 (1)	3 (1)	3 (1)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.4^{\dagger}$ (7)	$^{\dagger}1.2^{\dagger}$ (8)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5))	$^{\dagger}2.6^{\dagger}$ (5)	$^{\dagger}1.54^{\dagger}$(6)
CF_7	1.92 (2)	1.98 (1)	$^{\dagger}1.8^{\dagger}$ (4)	$^{\dagger}1.86^{\dagger}$ (3)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1^{\dagger}$ (7)	$^{\dagger}0.5^{\dagger}$ (8)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.41^{\dagger}$ (5)	$^{\dagger}1.07^{\dagger}$ (6)
CF_8	3 (1)	3 (1)	2.9 (4)	3 (1)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.4^{\dagger}$ (7)	$^{\dagger}1.5^{\dagger}$ (6)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.59^{\dagger}$ (5)	$^{\dagger}1.33^{\dagger}$ (8)
CF_9	3 (1)	3 (1)	3 (1)	3 (1)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.8^{\dagger}$ (6)	$^{\dagger}1.5^{\dagger}$ (7)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.9^{\dagger}$ (5)	$^{\dagger}1.42^{\dagger}$ (8)
CF_{10}	1.8† (3)	$^{\ddagger}2$ (1)	$^{\dagger}1.2^{\dagger}$ (5.5)	$^{\ddagger}2$ (1)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.2^{\dagger}$ (5.5)	$^{\dagger}1.1^{\dagger}$ (7)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.26^{\dagger}$ (4)	$^{\dagger}1^{\dagger}$ (8)
CF_{11}	3† (3)	$^{\ddagger}4$ (1)	2.9 (4)	4 (1)	$^{\dagger}0^{\dagger}$ (10)	$^{\dagger}1.3^{\dagger}$ (5)	$^{\dagger}0^{\dagger}$(10)	$^{\dagger}0^{\dagger}$(10)	$^{\dagger}0^{\dagger}$ (10)	$^{\dagger}0^{\dagger}$ (10)	$^{\dagger}0.68^{\dagger}$ (6)	$^{\dagger}0.32^{\dagger}$ (7)
CF_{12}	2.72† (3)	$^{\ddagger}2.94$(1)	$^{\dagger}2.4^{\dagger}$ (4)	$^{\ddagger}2.84^{\dagger}$ (2)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.7^{\dagger}$ (5.5)	$^{\dagger}1.6^{\dagger}$ (7)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.7^{\dagger}$ (5.5)	$^{\dagger}1.25^{\dagger}$ (8)
CF_{13}	2.8† (3)	$^{\ddagger}3.9$ (1)	$^{\dagger}2.4^{\dagger}$ (4)	$^{\ddagger}3.78^{\dagger}$ (2)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0.9^{\dagger}$ (7)	$^{\dagger}0.3^{\dagger}$ (8)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.39^{\dagger}$(6)	$^{\dagger}1.61^{\dagger}$ (5)
CF_{14}	1 (1)	1 (1)	1 (1)	1 (1)	$^{\dagger}0^{\dagger}$ (10.5)	1 (1)	1 (1)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	1 (1)	1 (1)
CF_{15}	4 (1)	4 (1)	$^{\dagger}3.8^{\dagger}$ (4)	4 (1)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}1.6^{\dagger}$ (7)	$^{\dagger}1.2^{\dagger}$ (8)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}0^{\dagger}$ (10.5)	$^{\dagger}2^{\dagger}$ (6)	$^{\dagger}2.1^{\dagger}$ (5)
Total ranks	30	15	49.5	21	150.5	95.5	101.5	157.5	157.5	157.5	78.5	98

B. Experimental Setup

The proposed algorithms are compared with a number of commonly used niching algorithms. In total, 15 different multimodal algorithms are considered in our experiments, which are frequently cited in literatures.

1) Self-CCDE: the crowding DE with the proposed clustering partition and self-adaptive strategy.
2) Self-CSDE: the speciation DE with the proposed clustering partition and self-adaptive strategy.
3) CCDE: the crowding DE with the proposed clustering partition alone.
4) CSDE: the speciation DE with the proposed clustering partition alone.
5) CDE [22]: the original crowding DE.
6) SDE [23]: the original speciation-based DE.
7) FER-PSO [36]: fitness-Euclidean distance ratio PSO.
8) SPSO [35]: speciation-based PSO.
9) r2pso [36]: a local best PSO with a ring topology, each member interacts with only its immediate member to its right.
10) r3pso [36]: a local best PSO with a ring topology, each member interacts with its immediate member on its left and right.
11) CMA [38]: niching covariance matrix adaptation evolution strategy.
12) SCMA [38]: CMA with self-adaptive niche radius.
13) NShDE [30]: the neighborhood-based sharing DE.
14) LIPS [41]: the locally informed PSO.
15) IWO-δ-GSO [40]: an invasive weed optimizer followed by the subregional modified group search optimizer algorithm.

The PSO and CMA parameters used in this paper are adopted from [38] and [41], while the DE parameters are the same as those used in [30]. The subpopulation size M is set to be five, and ten in the cases of $Np \leq 200$ and $Np > 200$, respectively.

C. Population Size and Maximal Number of Evaluations

In the experiment, a level of accuracy (typically $0 < \varepsilon < 1$), indicating how close the fitness values of the computed solutions to the known global peaks are, needs to be specified to compare different algorithms fairly. If the difference from a computed solution to a known global optimum is below ε, then the peak is considered to have been found and no other solutions within the niche radius (Euclidian distance $<$ niche radius) are counted. The level of accuracy (ε), niching radius (r), population size, and maximal number of function evaluations (FEs) allowed are listed in Table II. These settings apply to all compared algorithms. Different population size and different maximum numbers of FEs are allotted for different functions depending upon their degrees of complexity.

TABLE V

AVERAGE NUMBER OF GLOBAL PEAKS FOUND FOR TEST FUNCTION SET 2 WITH D=30 AND THE RESPECTIVE RANKS (IN PARENTHESES)

Fun	Self-CCDE	Self-CSDE	CCDE	CSDE	CDE [22]	SDE [23]	FER-PSO [36]	SPSO [36]	r2pso [36]	r3pso [36]	SCMA [38]	CMA[38]
CF_1	2 (1)	2 (1)	†0.9† (5)	†1.8† (3)	†0† (10)	†1.25† (4)	†0† (10)	†0† (10)	†0† (10)	†0† (10)	†0.28† (7)	†0.32† (6)
CF_2	1.6† (2)	‡1.8 (1)	†1.3† (5)	†1.5† (3)	†1† (8.5)	†1† (8.5)	†1.34† (4)	†0† (11)	†0† (11)	†0† (11)	†1.2† (7)	†1.25† (6)
CF_3	3.4† (2)	‡3.5 (1)	†2.6† (4)	†3.0† (3)	†0† (10.5)	†0.88† (5)	†0.35† (7)	†0† (10.5)	†0† (10.5)	†0† (10.5)	†0.6† (6)	†0.32† (8)
CF_4	1 (1)	1 (1)	†0.8† (3)	†0.6† (4)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)
CF_5	1 (1)	1 (1)	†0.7† (3.5)	†0.7† (3.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)	†0† (8.5)
CF_6	1 (1)	1 (1)	†0.6† (7)	1 (1)	†0† (10)	1 (1)	†0† (10)	†0† (10)	†0† (10)	†0† (10))	1 (1)	1 (1)
CF_7	1.52† (2)	‡1.7 (1)	†0.8† (6)	†1.4† (3)	†0† (9.5)	†1† (4.5)	†0† (9.5)	†0† (9.5)	†0† (9.5)	†0† (9.5)	†1† (4.5)	†0† (9.5)
CF_8	0.8† (4)	‡1 (1)	†0.2† (7)	‡1 (1)	†0† (10)	‡1 (1)	†0† (10)	†0† (10)	†0† (10)	†0† (10)	†0.54† (5)	†0.46† (6)
CF_9	1.4† (2)	‡1.5 (1)	†1† (4)	†1.1† (3)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0.6† (5)
CF_{10}	1 (1)	1 (1)	†0.7† (5)	1 (1)	†0† (9.5)	†0† (9.5)	†0.4† (6)	†0† (9.5)	†0† (9.5)	†0† (9.5)	1 (1)	†0† (9.5)
CF_{11}	1.7 (2)	1.72 (1)	†1.1† (4)	†1.2† (3)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0.24† (5)
CF_{12}	1.6† (2)	‡1.8 (1)	†1† (5.5)	†1.5† (3)	†0† (10)	†1† (5.5)	†0† (10)	†0† (10)	†0† (10)	†0† (10)	†1† (5.5)	†1† (5.5)
CF_{13}	1 (1)	1 (1)	†0.6† (4)	†0.8† (3)	†0† (9)	†0.2† (5)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)
CF_{14}	0.8† (3)	‡0.9 (2)	†0.5† (5)	†0.6† (4)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	†0† (9)	‡1‡ (1)
CF_{15}	1.5† (3)	‡1.7 (1)	†1.2† (4)	‡1.6† (2)	†0† (10.5)	†1† (6.5)	†1† (6.5)	†0† (10.5)	†0† (10.5)	†0† (10.5)	†1† (6.5)	†1† (6.5)
Total ranks	28	15	72	40.5	141.5	94.5	126	144	144	144	97	95

TABLE VI

SUCCESS RATES IN PERCENTAGE

Fun	Self-CCDE	Self-CSDE	CCDE	CSDE	CDE[22]	SDE [23]	FER-PSO [36]	SPSO [36]	r2pso [36]	r3pso [36]	SCMA [38]	CMA[38]
F_1	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	82 (10)	44 (12)	76 (11)	84 (9)	100 (1)	100 (1)
F_2	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	44 (12)	88 (11)	96 (10)	100 (1)	100 (1)
F_3	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	96 (6.5)	20 (9)	4 (12)	8 (10.5)	8 (10.5)	92 (8)	96 (6.5)
F_4	100 (1)	100 (1)	100 (1)	100 (1)	28 (10)	72 (9)	84 (8)	88 (6.5)	92 (5)	88 (6.5)	12 (12)	20 (11)
F_5	100 (1)	100 (1)	100 (1)	100 (1)	72 (12)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)
F_6	100 (1)	100 (1)	100 (1)	86 (7)	28 (10)	60 (9)	100 (1)	92 (5)	88 (6)	72 (8)	0 (12)	8 (11)
F_7	100 (1)	100 (1)	100 (1)	100 (1)	60 (12)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)
F_8	100 (1)	100 (1)	72 (5)	88 (3)	0 (11.5)	72 (5)	72 (5)	0 (11.5)	28 (9)	24 (10)	70 (7)	68 (8)
F_9	100 (1)	100 (1)	90 (7)	100 (1)	0 (11.5)	100 (1)	96 (6)	0 (11.5)	56 (10)	60 (9)	66 (8)	100 (1)
F_{10}	100 (1)	100 (1)	92 (5)	100 (1)	52 (9)	32 (10)	100 (1)	56 (8)	88 (6)	76 (7)	4 (12)	18 (11)
F_{11}	100 (1)	100 (1)	100 (1)	100 (1)	72 (6)	46 (9)	52 (8)	0 (12)	4 (10.5)	4 (10.5)	60 (7)	100 (1)
F_{12}	100 (1)	74 (3)	88 (2)	68 (5.5)	56 (10.5)	48 (12)	60 (7.5)	72 (4)	68 (5.5)	56 (10.5)	60 (7.5)	58 (9)
F_{13}	92 (1)	48 (3)	86 (2)	30 (4)	8 (5)	0 (9)	0 (9)	0 (9)	0 (9)	0 (9)	0 (9)	0 (9)
F_{14}	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
Total ranks	14	18	30	29.5	101.5	75.5	68.5	106.5	96.5	103	87.5	72.5

A large number of optima requires a larger population size and more function evaluations. The detailed setting for experiments can be found in [30].

D. Performance Measure

The performance of each of the multimodal optimizers is measured in terms of the following two criteria:

1) Success Rate: the percentage of runs in which all global peaks are successfully located within the budget of maximum FEs; and
2) Average number of optima found [39].

All performances are calculated and averaged over 50 independent runs. In order to show the significant differences between two algorithms, the Mann–Whitney–Wilcoxon rank sum test on the average numbers of peaks found by two competing algorithms at 5% significance level is also conducted [30], [40], [41]. The marks † and ‡ in the *left* indicate that self-CCDE is significantly better than and worse than the corresponding algorithm, respectively, while, the marks † and ‡ in the *right* indicate that self-CSDE is significantly better than and worse than the corresponding algorithm, respectively. If an entry has no such marking, it means the difference of two algorithms is not statistically significant.

V. EXPERIMENTAL RESULTS

A. Comparisons With Other State-of-the-Art Algorithms

This section presents the experiment results. On all the benchmark functions, all algorithms are run until all known

TABLE VII

AVERAGE NUMBER OF GLOBAL PEAKS FOUND FOR TEST FUNCTION SET 1 AND TEST FUNCTION SET 2 WITH D=10 AND THE RESPECTIVE RANKS (IN PARENTHESES)

Fun	Self-CCDE	Self-CSDE	LIPS [41]	NShDE [30]	IWO-δ-GSO [40]
F_1	1 (1)	1 (1)	NA	1 (1)	1 (1)
F_2	1 (1)	1 (1)	NA	1 (1)	1 (1)
F_3	2 (1)	2 (1)	NA	2 (1)	2 (1)
F_4	5 (1)	5 (1)	5 (1)	5 (1)	5 (1)
F_5	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
F_6	5 (1)	5 (1)	5 (1)	5 (1)	5 (1)
F_7	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
F_8	4 (1)	4 (1)	4 (1)	3.92 (4)	†3.88† (5)
F_9	2 (1)	2 (1)	2 (1)	2 (1)	†1.6† (5)
F_{10}	1 (1)	1 (1)	1 (1)	0.96 (5)	1 (1)
F_{11}	18 (1)	18 (1)	†17.82† (4)	18 (1)	NA
F_{12}	6 (1)‡	†5.89 (3)	NA	†5.88 (4)	6‡ (1)
F_{13}	35.94‡ (2)	†33 (3)	NA	35.96‡ (1)	†29.6† (4)
F_{14}	185.22‡ (1)	†118.86 (3)	NA	†179‡ (2)	†102.6† (4)
CF_1	5.6† (2)	‡6.9 (1)	†3.7† (3.5)	†3.7† (3.5)	†2† (5)
CF_2	4 (1)	4 (1)	†2.1† (4)	†2.8† (3)	†2† (5)
CF_3	6 (1)	6 (1)	†4† (3.5)	†4† (3.5)	†3.6† (5)
CF_4	5.2† (2)	‡5.6 (1)	†1.9† (4)	†4.5† (3)	†0† (5)
CF_5	5.5† (2)	‡6 (1)	†2.2† (4)	†3.6† (3)	†2† (5)
CF_6	3 (3)	3 (3)	‡3.7‡ (1)	3 (3)	†1.52† (5)
CF_7	1.92 (2)	1.98 (1)	1.4 (4)	1 (5)	1.8 (3)
CF_8	3 (1)	3 (1)	3 (1)	3 (1)	†1.3† (5)
CF_9	3 (1)	3 (1)	†2.4† (4)	3 (1)	†1.8† (5)
CF_{10}	1.8† (3)	‡2 (1)	‡2 (1)	†1† (4)	NA
CF_{11}	3† (3)	‡4 (1)	‡3.4† (2)	†2.2† (4)	NA
CF_{12}	2.72† (2)	‡2.94 (1)	†2.6† (3)	†2† (4)	NA
CF_{13}	2.8† (3)	‡3.9 (1)	‡3.6† (2)	†1† (4)	NA
CF_{14}	1 (1)	1 (1)	1 (1)	1 (1)	NA
CF_{15}	4 (1)	4 (1)	†3.8† (3)	†2.4† (4)	NA

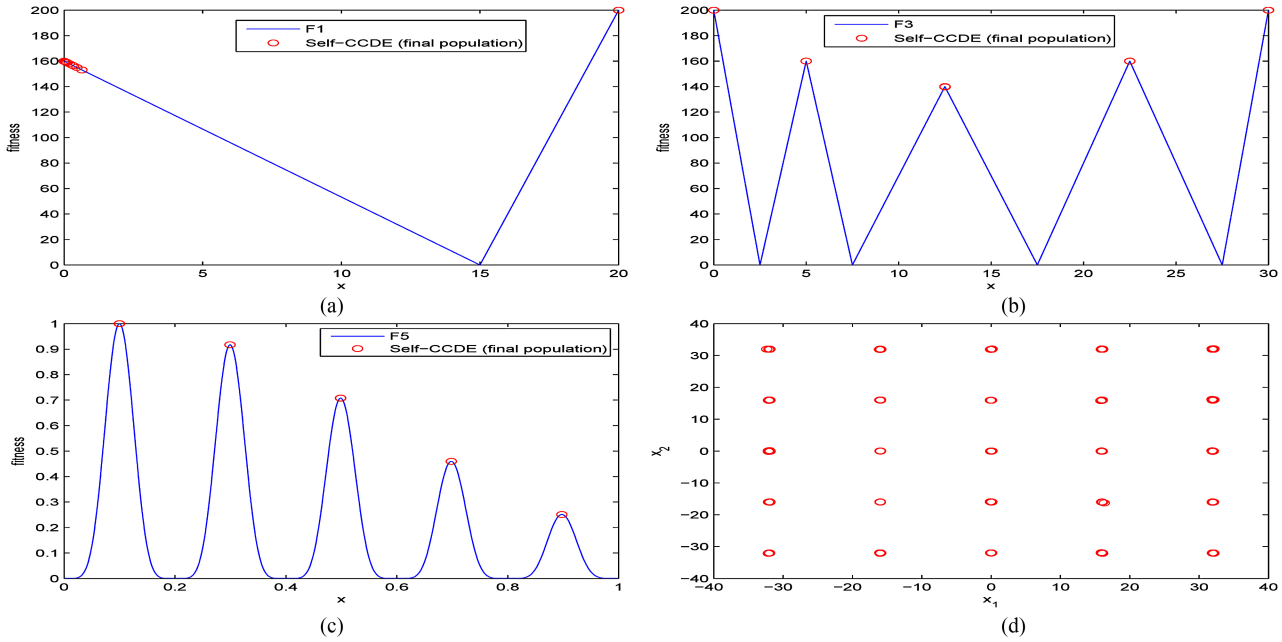

Fig. 1. Final population of self-CCDE for F_1, F_3, F_5, F_{10}. (a) F_1. (b) F_3. (c) F_5. (d) F_{10}.

TABLE VIII
AVERAGE CPU TIME TAKEN (IN SECONDS) FOR TEST FUNCTION SET 1 AND TEST FUNCTION SET 2 WITH D=10

Fun	Self-CCDE	Self-CSDE	CDE [22]	SDE [23]	SPSO [36]	FER-PSO [36]	r2pso [36]	r3pso [36]	SCMA [38]	CMA[38]
F_1 / F_2	0.89 / 1.02	1.18 / 1.38	1.04 / 1.16	1.34 / 1.52	1.38 / 1.32	2.64 / 2.70	0.88 / 0.92	0.94 / 0.94	2.72 / 2.28	1.86 / 2.15
F_3 / F_4	0.81 / 1.08	1.55 / 1.46	1.08 / 1.32	1.68 / 1.64	1.52 / 2.46	3.04 / 3.52	1.14 / 1.24	1.22 / 1.12	3.14 / 2.98	2.92 / 2.74
F_5 / F_6	0.80 / 1.11	1.15 / 1.39	0.92 / 1.24	1.28 / 1.54	1.36 / 1.78	2.08 / 2.14	0.86 / 1.12	0.84 / 1.08	1.62 / 1.84	1.84 / 1.93
F_7 / F_8	0.91 / 4.53	1.32 / 6.67	1.18 / 6.26	1.48 / 7.18	1.68 / 12.28	2.02 / 14.26	1.14 / 6.90	0.94 / 6.56	1.76 / 12.03	1.84 / 10.12
F_9 / F_{10}	5.20 / 10.21	7.65 / 15.02	8.34 / 20.82	9.24 / 24.14	14.60 / 27.34	13.62 / 30.24	9.74 / 22.39	8.57 / 18.38	13.82 / 32.58	12.88 / 28.48
F_{11} / F_{12}	22.58 / 17.21	17.45 / 18.27	29.44 / 23.84	28.36 / 26.84	33.08 / 22.88	36.28 / 18.58	26.38 / 12.62	24.74 / 13.26	35.28 / 30.94	32.74 / 24.84
F_{13} / F_{14}	26.25 / 37.27	40.21 / 45.89	30.86 / 35.86	36.72 / 39.72	32.48 / 46.72	41.76 / 57.62	30.64 / 33.72	28.24 / 30.32	39.62 / 54.36	38.40 / 52.74
CF_1 / CF_2	1686.5 / 2106.6	1702.8 / 2172.2	1544.3 / 1964.3	1590.6 / 2067.6	1677.4 / 2207.4	1798.3 / 2315.6	1286.2 / 1835.4	1270.8 / 1937.5	1805.2 / 2561.5	1739.5 / 2274.3
CF_3 / CF_4	1956.6 / 2521.4	2219.2 / 2326.2	1867.8 / 2410.5	2109.7 / 2205.6	2217.4 / 2338.4	2374.3 / 2563.9	2095.3 / 2292.4	1920.0 / 2274.3	2218.4 / 2351.4	1973.4 / 2017.4
CF_5 / CF_6	2605.6 / 2311.8	2468.8 / 2408.2	2473.2 / 2182.4	2316.3 / 2234.4	2415.2 / 2198.3	2579.2 / 2319.3	2271.3 / 2019.3	2310.4 / 2104.7	2561.8 / 2371.3	2583.0 / 2365.2
CF_7 / CF_8	2602.2 / 2752.2	2702.6 / 2783.5	2457.7 / 2615.7	2562.8 / 2627.6	2217.2 / 2516.8	2727.3 / 2724.1	2415.7 / 2561.3	2536.5 / 2497.3	2831.5 / 2804.6	2959.2 / 2718.3
CF_9 / CF_{10}	2712.3 / 2648.3	2733.2 / 2552.4	2547.2 / 2517.4	2617.4 / 2435.2	2416.2 / 2429.4	2918.4 / 2817.1	2416.2 / 2536.1	2415.3 / 2432.2	2531.7 / 2554.6	2251.3 / 2674.5
CF_{11} / CF_{12}	2924.2 / 3253.1	3127.6 / 3403.6	2819.4 / 3126.7	3019.4 / 3274.5	2823.4 / 3028.3	3126.5 / 3362.4	3001.2 / 3218.2	2964.2 / 3152.5	3182.6 / 3302.5	3042.5 / 3162.8
CF_{13} / CF_{14}	3500.5 / 3592.2	3512.3 / 3520.3	3328.6 / 3321.0	3398.7 / 3402.8	3525.8 / 3553.2	3627.1 / 3702.4	3062.1 / 3287.2	3071.4 / 3218.4	3502.6 / 3521.6	3416.7 / 3429.4
CF_{15}	4301.2	4382.6	4102.4	4226.4	4163.0	4372.2	3292.4	3282.3	4328.5	4029.7

peaks are found or the maximum number of FEs is exhausted. The average number of global peaks detected by each algorithm on test functions is summarized in Table III. Tables IV and V present the comparative results of 12 algorithms on scalable benchmark functions (i.e., CF_1 to CF_{15} with dimension of decision space equal to 10 and 30, respectively) in terms of the mean number of global peaks found. For clarity, the ranks of each algorithm among 12 competitors are shown in parentheses while the total ranks (summation of all the individual ranks) are listed in the last row of the table. For instance, self-CCDE can track in average 185.22 (out of 216) peaks for F_{14} in Table III and ranks one among 12 algorithms considered. It can be seen from Tables III, IV, and V that the proposed framework performs better than the other competitors. For these 44 cases, self-CCDE and self-CSDE rank on top two in most cases. If we focus on observing the DE variants, we can derive the following observation.

1) Compared to CDE, self-CCDE significantly improves performance on 41 cases out of 44, while the remaining three cases perform equally as both methods are able to locate all optima for all runs. Compared to SDE, self-CSDE improves results on 36 cases, while the remaining eight cases perform the same as both methods are able to locate all optima for all runs.
2) CCDE and CSDE are superior to CDE and SDE on 41 and 36 cases, respectively, while in the remaining three and eight cases, respectively, they perform equally.
3) Self-CCDE and self-CSDE are always better than CCDE and CSDE on 29 and 21 cases, respectively, and show the similar performance on the remaining cases.

Please note Mann–Whitney–Wilcoxon rank sum test is also performed here to compare self-CCDE and self-CSDE with other competitive algorithms. For example, in Table III, the cell in the fourth row (i.e., F_4) and the fifth column (i.e., CDE [22]) shows that the average number of global peak found is 3.84 over 50 independent runs. It ranks the tenth among the 12 chosen competing algorithms (i.e., behind self-CCDE, self-CSDE, CCDE, CSDE, SPSO, FER-PSO, r3pso, SDE, and r2pso in that order). The mark † on the left and on the right indicates that both self-CCDE and self-CSDE

TABLE IX
AVERAGE OF OPTIMA FOUND IN LOCATING BOTH GLOBAL AND LOCAL PEAKS

Fun	Self-CCDE	Self-CSDE	CCDE	CSDE	CDE [22]	SDE [23]	FER-PSO [36]	SPSO [36]	r2pso [36]	r3pso [36]	SCMA [38]	CMA[38]
F_1	2 (1)	2 (1)	2 (1)	2 (1)	2 (1)	†1.84† (8)	†1.48† (10.5)	†1.44† (12)	†1.72† (9)	†1.48† (10.5)	2 (1)	2 (1)
F_2	2 (1)	2 (1)	2 (1)	2 (1)	2 (1)	†1.68† (10)	†1.88† (8)	†1.72† (9)	†1.36† (11)	†1.24† (12)	2 (1)	2 (1)
F_3	5 (1)	5 (1)	5 (1)	†4.22†(5)	†4.44† (4)	†3.04† (8)	†0.64† (11)	†3.08† (7)	†0.8† (10)	†0.4† (12)	†2† (9)	†3.18† (6)
F_5	5 (1)	5 (1)	5 (1)	†4.86† (5)	†4.28† (6)	†1.52† (7)	†1† (9)	5 (1)	†1† (9)	†1† (9)	†0.52† (12)	†0.64† (11)
F_{10}	25 (1)	25 (1)	25 (1)	25 (1)	†12.5† (8)	†1.32† (11)	†5.16† (9)	†24.9† (5)	†24.2† (7)	†24.3† (6)	†0.86† (12)	†1.5† (10)
Total ranks	5	5	5	13	20	44	47.5	34	46	49.5	35	29

TABLE X
SUCCESS RATE IN LOCATING BOTH GLOBAL AND LOCAL PEAKS

Fun	Self-CCDE	Self-CSDE	CCDE	CSDE	CDE [22]	SDE [23]	FER-PSO [36]	SPSO [36]	r2pso [36]	r3pso [36]	SCMA [38]	CMA[38]
F_1	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	84 (8)	64 (10)	44 (12)	72 (9)	56 (11)	100 (1)	100 (1)
F_2	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	68 (10)	88 (8)	72 (9)	56 (11)	32 (12)	100 (1)	100 (1)
F_3	100 (1)	100 (1)	100 (1)	42 (5)	44 (4)	7 (7)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	8 (6)
F_5	100 (1)	100 (1)	100 (1)	88 (5)	48 (6)	0 (9.5)	0 (9.5)	100 (1)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)
F_{10}	100 (1)	100 (1)	100 (1)	100 (1)	0 (10)	0 (10)	0 (10)	92 (5)	60 (6)	52 (7)	0 (10)	0 (10)
Total ranks	5	5	5	13	22	44.5	47.5	37	45.5	49.5	31.5	27.5

TABLE XI
PEAK ACCURACY OF TEST FUNCTIONS

Fun	Self-CCDE	Self-CSDE	IWO-δ-GSO [40]	CDE [22]	SDE [23]	FER-PSO [36]	SPSO [36]	r2pso [36]	r3pso [36]	r2psolhc [36]	r3psolhc [36]
F_1	0 (1)	0 (1)	5.46e-20 (3)	9.46e-08 (5)	1.23e-08 (4)	5.24e-02 (8)	8.74e-02 (10)	4.65e-02 (7)	3.54e-02 (6)	9.78e-02 (11)	8.68e-02 (9)
F_2	0 (1)	0 (1)	3.47e-18 (3)	8.76e-06 (5)	3.43e-07 (4)	9.65e-04 (6)	9.45e-02 (11)	1.46e-02 (8)	1.43e-02 (7)	5.23e-02 (10)	4.54e-02 (9)
F_3	0 (1)	0 (1)	2.71e-17 (3)	9.76e-05 (5)	5.73e-05 (4)	4.34e-02 (6)	9.54e-01 (11)	9.89e-02 (9)	7.93e-02 (8)	5.52e-01 (10)	7.64e-02 (7)
F_4	1.80e-10 (2)	3.97e-10 (3)	5.11e-20 (1)	7.43e-05 (11)	9.53e-07 (10)	5.65e-07 (9)	3.12e-07 (8)	9.94e-08 (6)	2.24e-07 (7)	5.43e-09 (4)	9.43e-08 (5)
F_5	5.23e-15 (3)	6.66e-16 (2)	1.02e-20 (1)	9.43e-06 (11)	4.03e-09 (5)	8.34e-09 (6)	2.16e-09 (4)	1.54e-07 (7)	9.61e-07 (10)	8.58e-07 (9)	5.36e-07 (8)
F_6	8.19e-10 (2)	7.03e-11 (1)	9.71e-10 (3)	5.39e-05 (11)	8.27e-07 (10)	5.45e-09 (4)	9.58e-08 (7)	4.32e-07 (8)	8.79e-08 (6)	8.05e-07 (9)	7.54e-08 (5)
F_7	9.67e-10 (2)	3.40e-10 (1)	1.78e-07 (3)	8.97e-05 (11)	4.51e-07 (7)	7.41e-07 (8)	2.98e-07 (4)	4.35e-07 (6)	9.29e-07 (9)	3.43e-07 (5)	9.88e-07 (10)
F_8	8.02e-07 (2)	2.62e-07 (1)	2.76e-04 (3)	4.27e-02 (10)	8.57e-04 (4)	8.69e-04 (5)	5.21e-02 (11)	4.38e-03 (6)	5.63e-03 (7)	6.12e-03 (8)	9.19e-03 (9)
F_9	5.71e-10 (2)	4.70e-12 (1)	1.75e-05 (5)	3.42e-04 (10)	5.33e-08 (3)	7.38e-08 (4)	3.58e-04 (11)	5.36e-05 (7)	6.92e-05 (8)	4.33e-05 (6)	8.28e-05 (9)
F_{10}	2.99e-10 (1)	8.54e-10 (2)	7.66e-08 (3)	3.96e-03 (10)	9.92e-02 (11)	5.50e-06 (4)	9.77e-04 (9)	1.57e-05 (5)	8.31e-05 (6)	9.43e-05 (7)	6.87e-04 (8)
F_{12}	4.38e-08 (1)	8.38e-08 (2)	5.84e-08 (3)	5.42e-04 (10)	8.33e-04 (11)	4.35e-04 (8)	1.23e-04 (4)	1.78e-04 (5)	3.87e-04 (7)	2.54e-04 (6)	4.56e-04 (9)
F_{13}	4.42e-05 (1)	4.43e-05 (2)	3.77e-03 (4)	9.87e-04 (3)	9.85e-03 (11)	9.69e-03 (10)	8.45e-03 (8)	7.98e-03 (6)	8.25e-03 (7)	8.52e-03 (9)	6.32e-03 (5)
F_{14}	4.43e-04 (1)	8.12e-04 (2)	1.14e-01 (4)	9.23e-02 (3)	7.89e-01 (7)	5.95e-01 (5)	7.86e-01 (6)	9.53e-01 (10)	8.45e-01 (9)	9.68e-01 (11)	8.12e-01 (8)
Total ranks	20	20	39	105	91	83	104	90	97	105	101

perform significantly better than CDE statistically, respectively.

In order to compare the reliability of different algorithms, more experimental results are made and compared in Table VI. The results given are the success rates for all the algorithms on test functions $F_1 - F_{14}$. Since the 15 composition functions (CF_1 to CF_{15}) have far more complicated fitness landscapes as opposed to the basic functions, no algorithm except the proposed algorithms could achieve a nonzero success rate at least for some of them. For this reason, the results for the composite functions have not been reported. As can been seen from Table VI, self-CCDE achieves a much higher success rate than other algorithms on all of the 14 test functions. By comparing the results of test function sets 1 and 2, we can conclude that as the complexity of the problems increases, the multipopulation strategy and the self-adaptive parameter control technique can work together to easily outperform all the competing state-of-the-art algorithms. The experiment results and comparisons also verify that the algorithm with the two operators performs better than the algorithm with either or neither of them.

Further experiment results are listed in Table VII. Self-CCDE and self-CSDE are compared with LIPS [41], NShDE [30], and IWO-δ-GSO [40]. The results of the compared algorithms are all derived directly from their corresponding references. "NA" represents that the results are not available in the corresponding reference. It is clear that self-CCDE and self-CSDE work well and achieve better performance than three competitive algorithms in most of the cases. In addition, Mann–Whitney–Wilcoxon rank sum test is also made here to compare self-CCDE and self-CSDE with respect to LIPS [41], NShDE [30], and IWO-δ-GSO [40]. For example, in Table VII, the cell in the 11th row (i.e., for F_{11}) and the third column (i.e., LIPS [41]) shows that the average number of global peak found is 17.82 over 50 independent runs. It ranks the fourth among the five chosen competing algorithms (i.e., behind self-CCDE, self-CSDE, and NShDE). The mark † on the left and on the right indicates that self-CCDE and self-CSDE perform significantly better than LIPS statistically.

The average CPU time (in seconds) of each algorithm on test function sets 1 and 2 is reported in Table VIII. In this table, the CPU time is recorded after each algorithm is run until all known peaks are found or the maximum number of FEs is exhausted. Table VIII indicates that, for $F_1 - F_{14}$, the CPU time consumed by self-CCDE and self-CSDE is lower than CDE and SDE, respectively. It is because that self-CCDE and self-CSDE are able to find all known peaks before they reach the maximum number of FEs on almost all test functions considered. Thus, the actual computation time of self-CCDE and self-CSDE can be reduced in relative to CDE and SDE.

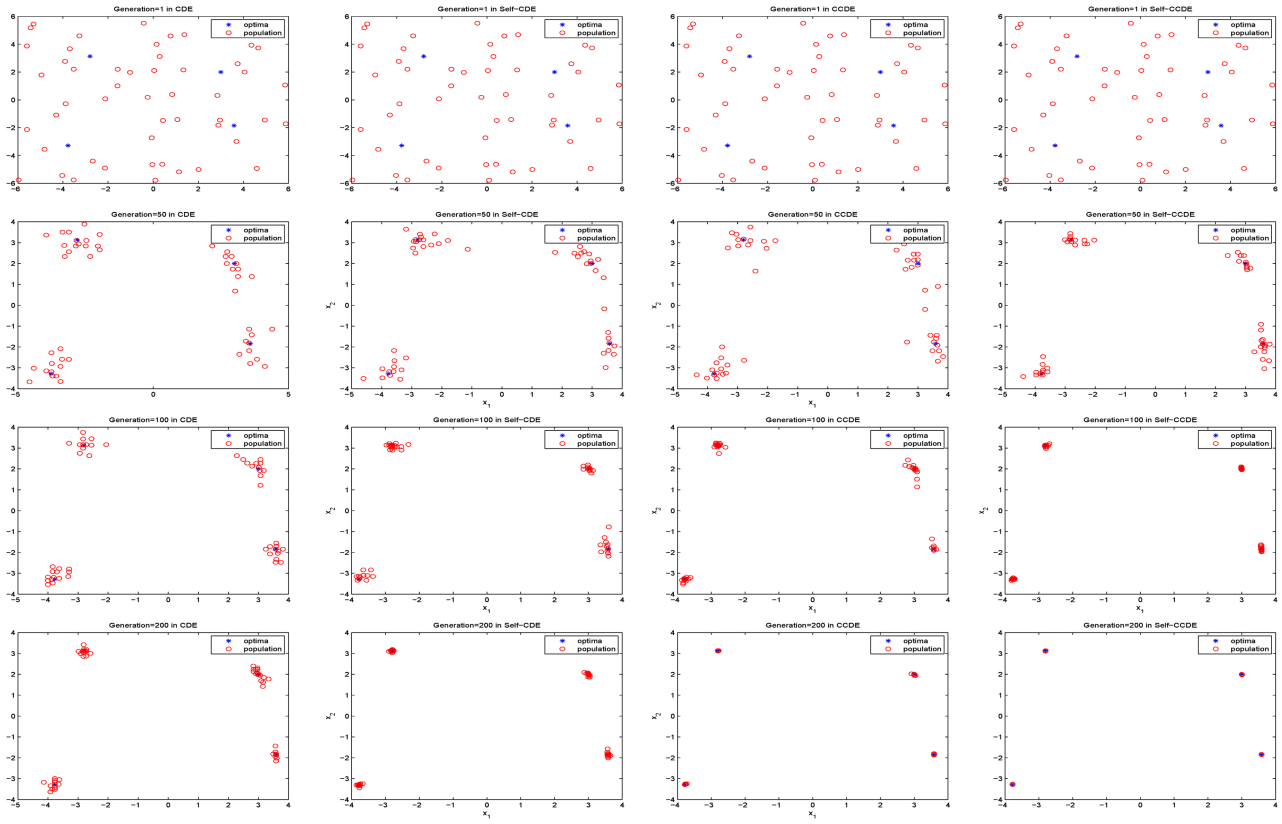

Fig. 2. Distributions of population of CDE, self-CDE, CCDE, and self-CCDE on different stages (F_8).

Compared to SPSO, FER-PSO, r2pso, r3pso, SCMA, and CMA, self-CCDE's computational time remains lower on test function set 1. However, for $CF_1 - CF_{15}$, no algorithm could reliably track all known peaks within the maximum number of FEs in a given run, and only self-CCDE and self-CSDE could track more peaks. However, owing to additional operators used in self-CCDE and self-CSDE, the computation time of self-CCDE and self-CSDE is slightly increased compared to CDE and SDE, respectively.

B. Locating Local Optima

For an optimization problem with multiple global optima as well as local optima, there is a need to locate all global optima and local optima. So a good niching algorithm should have the property of locating both global and local optima. In order to test the ability of locating local optima, five test functions (F_1, F_2, F_3, F_5, and F_{10}) from test set 1 are used. The results are shown in Tables IX and X. It can be observed that with the multipopulation strategy and the self-adaptive parameter control technique, the ability of locating both global and local optima is greatly improved. To give a clearer view, the final population of self-CCDE is also plotted (Fig. 1) for F_1, F_3, F_5, and F_{10}. Note that the population size and maximum number of function evaluations are set to 500 and 1 00 000 for F_{10}. The settings of remaining parameters are kept unchanged.

C. Advantage of the Multipopulation Strategy and the Self-Adaptive Parameter Control

To show the advantage of the multipopulation strategy and the self-adaptive parameter control technique, self-CCDE is compared with the original CDE, self-CDE (self-adaptive parameter control with CDE) and CCDE. The distributions of population of the four algorithms at different iterations are plotted in Fig. 2. As can be seen from the plots, for CDE, the distribution of population at iteration 200 is worse than self-CDE and CCDE at iteration 100, which implies that both the self-adaptive parameter control technique and the multi-population strategy have positive effect on the performance of the algorithm. On the other hand, self-CCDE converges even faster than self-CDE and CCDE, which verifies that the two operators can work together to improve the performance of CDE rather than contradict with each other.

D. Maintaining the Identified Optima

As stated in [41], a good niching algorithm must be able to locate global optima and maintain them until the stop of algorithm, with respect to population size. Maintaining found optima is important for an effective and stable multimodal optimization algorithm. Self-CCDE is able to find and maintain the identified optima until the end of the run. This is because the proposed algorithm uses the multipopulation strategy. Once a niche is formed around one global peak, the algorithm will continue to search better solutions within the same niche. Only when a better solution is found within the niche, it replaces the current found best solution. Fig. 3 shows the niching behavior of self-CCDE on F_4. From Fig. 3, we can see that self-CCDE is able to develop stable niches around the global peaks.

E. Comparison With Results Reported in [40]

In order to further demonstrate the superior performance of self-CCDE and self-CSDE, both are compared with the results

TABLE XII
DISTANCE ACCURACY OF TEST FUNCTIONS

Fun	Self-CCDE	Self-CSDE	IWO-δ-GSO [40]	CDE [22]	SDE [23]	FER-PSO [36]	SPSO [36]	r2pso [36]	r3pso [36]	r2psolhc [36]	r3psolhc [36]
F_1	0 (1)	0 (1)	5.46e-20 (3)	9.46e-08 (5)	1.23e-08 (4)	5.24e-02 (8)	8.74e-02 (10)	4.65e-02 (7)	3.54e-02 (6)	9.78e-02 (11)	8.68e-02 (9)
F_2	0 (1)	0 (1)	3.47e-18 (3)	8.76e-06 (5)	3.43e-07 (4)	9.65e-04 (6)	9.45e-02 (11)	1.46e-02 (8)	1.43e-02 (7)	5.23e-02 (10)	4.54e-02 (9)
F_3	0 (1)	0 (1)	2.71e-17 (3)	9.76e-05 (5)	5.73e-05 (4)	4.34e-02 (6)	9.54e-01 (11)	9.89e-02 (9)	7.93e-02 (8)	5.52e-01 (10)	7.64e-02 (7)
F_4	1.80e-10 (2)	3.97e-10 (3)	5.11e-20 (1)	7.43e-05 (11)	9.53e-07 (10)	5.65e-07 (9)	3.12e-07 (8)	9.94e-08 (6)	2.24e-07 (7)	5.43e-09 (4)	9.43e-08 (5)
F_5	5.23e-15 (3)	6.66e-16 (2)	1.02e-20 (1)	9.43e-06 (11)	4.03e-09 (5)	8.34e-09 (6)	2.16e-09 (4)	1.54e-07 (7)	9.61e-07 (10)	8.58e-07 (9)	5.36e-07 (8)
F_6	8.19e-10 (2)	7.03e-11 (1)	9.71e-10 (3)	5.39e-05 (11)	8.27e-07 (10)	5.45e-09 (4)	9.58e-08 (7)	4.32e-07 (8)	8.79e-08 (6)	8.05e-07 (9)	7.54e-08 (5)
F_7	9.67e-10 (2)	3.40e-10 (1)	1.78e-07 (3)	8.97e-05 (11)	4.51e-07 (7)	7.41e-07 (8)	2.98e-07 (4)	4.35e-07 (6)	9.29e-07 (9)	3.43e-07 (5)	9.88e-07 (10)
F_8	8.02e-07 (2)	2.62e-07 (1)	2.76e-04 (3)	4.27e-02 (10)	8.57e-04 (4)	8.69e-04 (5)	5.21e-02 (11)	4.38e-03 (6)	5.63e-03 (7)	6.12e-03 (8)	9.19e-03 (9)
F_9	5.71e-10 (2)	4.70e-12 (1)	1.75e-05 (5)	3.42e-04 (10)	5.33e-08 (3)	7.38e-08 (4)	3.58e-04 (11)	5.36e-05 (7)	6.92e-05 (8)	4.33e-05 (6)	8.28e-05 (9)
F_{10}	2.99e-10 (1)	8.54e-10 (2)	7.66e-08 (3)	3.96e-03 (10)	9.92e-02 (11)	5.50e-06 (4)	9.77e-04 (9)	1.57e-05 (5)	8.31e-05 (6)	9.43e-05 (7)	6.87e-04 (8)
F_{12}	4.38e-08 (1)	8.38e-08 (2)	5.84e-08 (3)	5.42e-04 (10)	8.33e-04 (11)	4.35e-04 (8)	1.23e-04 (4)	1.78e-04 (5)	3.87e-04 (7)	2.54e-04 (6)	4.56e-04 (9)
F_{13}	4.42e-05 (1)	4.43e-05 (2)	3.77e-03 (4)	9.87e-04 (3)	9.85e-03 (11)	9.69e-03 (10)	8.45e-03 (8)	7.98e-03 (6)	8.25e-03 (7)	8.52e-03 (9)	6.32e-03 (5)
F_{14}	4.43e-04 (1)	8.12e-04 (2)	1.14e-01 (4)	9.23e-02 (3)	7.89e-01 (7)	5.95e-01 (5)	7.86e-01 (6)	9.53e-01 (10)	8.45e-01 (9)	9.68e-01 (11)	8.12e-01 (8)
Total ranks	20	20	39	105	91	83	104	90	97	105	101

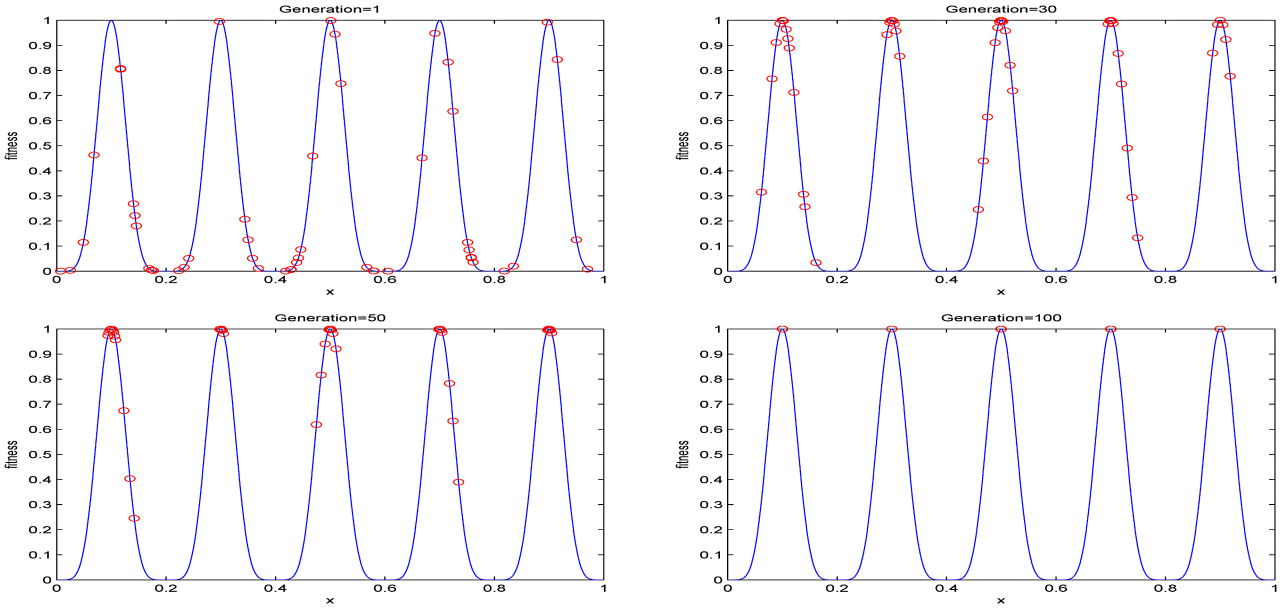

Fig. 3. Distribution of population over function evaluations (F_4).

reported in one of the latest multimodal optimization works [40]. The results of the other compared algorithms are directly taken from [40]. More details about this experiment can be found in [40]. The assessment criteria are also adopted from [40], which are listed as below.

1) Peak Accuracy: For each desired $peak_i, (i = 1, 2, ..., \#peaks)$ to be located, the closest individual X in the population is taken and absolute difference in objective values is calculated. If the objective value of individual X is denoted by $f(X)$, the peak accuracy is calculated using

$$peak \quad accuracy = \sum_{i=1}^{\#peaks} \frac{\mid f(peak_i) - f(X) \mid}{\#peaks}. \qquad (8)$$

2) Distance Accuracy: Peak accuracy may lead to erroneous results, if the peaks are close to each other or with identical height. The distance accuracy is used to avoid this error. It is calculated the same way as peak accuracy, with the only change that the fitness values are replaced by the Euclidean distance. The performance

TABLE XIII
EFFECT OF THE SUBPOPULATION SIZE M ON SELF-CCDE

Test Functions	Population Size	M = 5	M = 10	M = 20	M = 25
F_1	50	1	1	-	1
F_2	50	1	1	-	1
F_3	50	1	1	-	0.95
F_4	50	1	1	-	0.92
F_5	50	1	0.97	-	0.91
F_6	50	1	0.96	-	0.93
F_7	50	1	1	-	0.90
F_8	50	1	1	-	0.92
F_9	50	1	0.97	-	0.94
F_{10}	50	1	1	-	0.86
F_{11}	250	0.94	1	-	0.90
F_{12}	100	1	0.98	0.96	0.92
F_{13}	500	0.96	0.99	0.98	0.94
F_{14}	1000	0.85	0.86	0.82	0.78

results are shown in Tables XI and XII. The ranks of each algorithm are shown in the parentheses. From the results, we can see that the proposed algorithms rank on the top for almost all test functions in terms of

peak accuracy and distance accuracy. It further indicates that the proposed algorithms exhibit a good exploitative behavior that facilitates convergence toward different global and local optima. Please note function F_{11} is a 2-D inverted Shubert function, which is not reported in [40]. Thus, the experimental results of function F_{11} are not included in Tables XI and XII. Test functions F_{12}, F_{13}, and F_{14} are 1-D, 2-D, and 3-D inverted Vincent functions, respectively, which are corresponding to functions F_{11}, F_{12}, and F_{13} in [40].

F. Effect of the Population Size and Subpopulation Size

If the population size is smaller, the algorithm has a poor ability to explore the whole search space. The algorithm thus will miss some of the peaks. On the other hand, if the population size is larger, the function evaluations at each generation will increase. Under the same budget of function evaluations, the algorithm may not converge to the peaks. The population size in this paper follows the guideline given in [30] and [41]. It can be clearly observed that when the population size is under this setting, we obtain better results on all of the test functions.

Next, we investigate the impact of the subpopulation size M on self-CCDE. The maximum number of function evaluations is listed in Table II. '-' represents that the population size Np cannot be evenly divided by the chosen subpopulation size M. Self-CCDE runs 50 times on each test function, and the average numbers of peak ratio (the percentage of successfully located peaks) are reported in Table XIII. In general, a smaller subpopulation size will produce a better diversity for the population, while a larger subpopulation size is beneficial for convergence. From Table XIII, we can observe that M can influence the results. For test functions $F_1 - F_{10}$ and F_{12}, the average number of peaks found drops as the subpopulation size increases in most cases. Although algorithm with a larger subpopulation size will increase its convergence speed, it suffers at the same time with a poor diversity and often misses some of the peaks. On the other hand, F_{11} and $F_{13} - F_{14}$ are complex functions, which have many peaks and finding any peak is a challenging task. Therefore, the convergence is considered more important than diversity. How to choose a proper subpopulation size depends on the functional landscape of the problem and the expected quality of the solutions to be found. If the diversity is a primary concern, a smaller subpopulation size should be used. Otherwise a larger subpopulation size should be selected. In this experiment, we observe an interesting result that for $F_1 - F_{10}$ and F_{12} whose population sizes are within $Np \leq 200$, we obtain better results at $M = 5$, while for the other test functions whose population sizes are beyond $Np > 200$, we obtain better results at $M = 10$. Therefore, subpopulation size is selected based on the heuristics found in the experiment.

VI. Conclusion

In this paper, we developed a cluster-based differential evolution with self-adaptive strategy for solving multimodal optimization problems. The clustering partition is used to divide the whole population into subpopulations, with the aim of driving individuals to search in different subregions to obtain potential multiple global and local optima. In order to further improve the performance of the algorithm, a self-adaptive parameter control technique is incorporated into the design. In this paper, the multipopulation strategy and the self-adaptive parameter control technique are applied to two versions of DE, CDE, and SDE, yielding self-CCDE and self-CCDE, respectively. Comprehensive experimental tests have been conducted on two different sets of benchmark functions. The experiment results show the effectiveness and the efficiency of the multipopulation strategy and the self-adaptive parameter control technique.

However, the subpopulation size M affects the performance of the algorithm to some extent. How to design an adaptive strategy to control M through the evolution process calls for future work. Future research may also focus on testing the performance of the algorithm on much more massively multimodal optimization problems with high dimensionality and constraints.

References

[1] S. Das, S. Maity, B. Y. Qu, and P. N. Suganthan, "Real-parameter evolutionary multimodal optimization: A survey of the state-of-the-art," *Swarm Evol. Comput*. vol. 1, no. 2, pp. 71–88, Jun. 2011.

[2] H. F. Wang, I. Moon, S. X. Yang, and D. W. Wang, "A memetic particle swarm optimization algorithm for multimodal optimization problems," *Inf. Sci.*, vol. 197, no.1, pp. 38–52, 2012.

[3] S. Roy, S. M. Islam, S. Das, and S. Ghosh, "Multimodal optimization by artificial weed colonies enhanced with localized group search optimizers," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 27–46, 2013.

[4] M. Q. Li, D. Lin, and J. S. Kou, "A hybrid niching PSO enhanced with recombination-replacement crowding strategy for multimodal function optimization," *Appl. Soft Comput.*, vol. 12, no. 3, pp. 975–987, 2012.

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

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

[7] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[8] W. Y. Gong, Z. H. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 397–413, Apr. 2011.

[9] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 482–500, Apr. 2012.

[10] S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 107–124, Feb. 2012.

[11] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 634–647, Apr. 2013.

[12] W. Y. Gong and Z. H. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Trans. Cybern.*, 2013.

[13] Y. Q. Cai and J. H. Wang, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Trans. Cybern.*, vol. 41, no. 2, pp. 397–413, Apr. 2011.

[14] B. Rigling and F. Moore, "Exploitation of subpopulations in evolutionary strategies for improved numerical optimization," in *Proc. 11th Midwest Artif. Intell. Cogn. Sci. Conf.*, 1999, pp. 80–88.

[15] J. Rumpler and F. Moore, "Automatic selection of subpopulations and minimal spanning distances for improved numerical optimization," in *Proc. Congr. Evol. Comput.*, 2001, pp. 38–43.
[16] D. Zaharie, "A multipopulation differential evolution algorithm for multimodal optimization," in *Proc. 10th Mendel Int. Conf. Soft Comput.*, Jun. 2004, pp. 17–22.
[17] Z. Hendershot, "A differential evolution algorithm for automatically discovering multiple global optima in multidimensional discontinuous spaces," in *Proc. 15th Midwest Artif. Intell. Cogn. Sci. Conf.*, Apr. 2004, pp. 92–97.
[18] D. Zaharie, "Extensions of differential evolution algorithms for multimodal optimization," in *Proc. SYNASC*, 2004, pp. 523–534.
[19] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.
[20] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[21] 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.
[22] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2004, pp. 1382–1389.
[23] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," in *Proc. Conf. Genetic Evol. Comput.*, 2005, pp. 873–880.
[24] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 881–897, Jun. 2013.
[25] Y. Q. Cai, J. H. Wang, and J. Yin, "Learning-enhanced differential evolution for numerical optimization," *Soft Comput.*, vol. 16, no. 2, pp. 303–330, 2012.
[26] Z. H. Cai, W. Y. Gong, C. X. Ling, and H. Zhang, "A clustering-based differential evolution for global optimization," *Appl. Soft Comput.*, vol. 11, no.1, pp. 1363–1379, 2011.
[27] Y. J. Wang, J. S. Zhang, and G. Y. Zhang, "A dynamic clustering based differential evolution algorithm for global optimization," *Eur. J. Oper. Res.*, vol. 183, no. 1, pp. 56–73, 2007.
[28] J. Rönkkönen and J. Lampinen, "On determining multiple global optima by differential evolution," in *Proc. Evol. Deterministic Meth. Design, Optimiz. Control*, Jun. 2007, pp. 146–151.
[29] K. C. Wong, C. H. Wuc, R. K. P. Mokd, C. Penge, and Z. Zhang, "Evolutionary multimodal optimization using the principle of locality," *Inf. Sci.*, vol. 194, pp. 138–170, Jul. 2012.
[30] 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.
[31] S. Roy, S. K. M. Islam, S. Das, S. Ghosh, and A. V. Vasilakos, "A simulated weed colony system with sub-regional differential evolution for multimodal optimization," *Engineering Optimization*, vol. 45, no. 4, pp. 459–481, 2013.
[32] 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.
[33] K. Deb and A. Saha, "Multimodal optimization using a bi-objective evolutionary algorithm," IIT Kanpur, Kanpur, India, KanGAL Rep. 2009006, Dec. 2009.
[34] A. Basak, S. Das and K.C. Tan, "A bi-objective differential evolution algorithm enhanced with mean distance based selection for multimodal optimization," *IEEE Trans. Evol. Comput.*, Early Access, 2013 with DOI 10.1109/TEVC.2012.2231685
[35] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proc. Genetic Evol. Comput. Conf.*, vol. 3102. 2004, pp. 105–116.
[36] 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.
[37] B. Y. Qu and P. N. Suganthan, "Novel multimodal problems and differential evolution with ensemble of restricted tournament selection," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 1–7.
[38] O. M. Shir, M. Emmerich, and T. Back, "Adaptive niche radii and niche shapes approaches for niching with the CMA-ES," *Evol. Comput.*, vol. 18, no. 1, pp. 97–126, 2010.
[39] J. Gan, and K. Warwick, "A variable radius niching technique for speciation in genetic algorithms," in *Proc. GECCO*, 2000, pp. 96–103.
[40] S. Roya, S. M. Islama, S. Dasb, and S. Ghosha, "Multimodal optimization by artificial weed colonies enhanced with localized group search optimizers," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 27–46, Jan. 2013.
[41] B. Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multi-modal optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 387–402, Jun. 2013.

Weifeng Gao received the B.S. degree in applied mathematics from the School of Science, Xidian University, Xi'an, China, in 2008, where he is currently pursuing the Ph.D. degree.

His current research interests include evolutionary algorithms and their applications in the real world.

Gary G. Yen (S'87–M'88–SM'97–F'09) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1992.

He is currently a Professor with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA. His current research interests include intelligent control, computational intelligence, conditional health monitoring, signal processing, and their industrial/defense applications.

Dr. Yen was an Associate Editor of the IEEE Control Systems Magazine, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, *Automatica*, *Mechantronics*, the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, PART A AND PART B, and the IEEE TRANSACTIONS ON NEURAL NETWORKS. He is currently an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and the *International Journal of Swarm Intelligence Research*. He served as the General Chair for the 2003 IEEE International Symposium on Intelligent Control held in Houston, TX and 2006 IEEE World Congress on Computational Intelligence held in Vancouver, Canada. He was a Vice President for the Technical Activities in 2005-2006 and President in 2010-2011 for the IEEE Computational Intelligence Society and is the Founding Editor-in-Chief of the IEEE Computational Intelligence Magazine 2006-2009. In 2011, he received the Andrew P. Sage Best Transactions Paper award from the IEEE Systems, Man and Cybernetics Society. In 2013, he received Meritorious Service award from the IEEE Computational Intelligence Society. He is a fellow of IET.

Sanyang Liu received the Ph.D. degree in computational mathematics from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 1989.

He is currently a Professor and the Dean of School of Science, Xidian University, Xi'an, Shaanxi, China. His current research interests include nonlinear optimization, combinatorial optimization, network optimization, system reliability, and so on.