Co-Evolutionary Niching Differential Evolution Algorithm for Global Optimization

LE YAN¹, JIANJUN CHEN², QI LI³, JIAFA MAO⁴, AND WEI GUO SHENG³, (Member, IEEE)

¹School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China
²Keyi College of Zhejiang Sci-Tech University, Hangzhou, Shangyu 312369, China
³School of Information Science and Engineering, Hangzhou Normal University, Hangzhou 311121, China
⁴School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China

Corresponding author: Weiguo Sheng (w.sheng@ieee.org)

This work was supported in part by the National Natural Science Foundation of China under Grant 61873082, Grant 62003121, and Grant 62176237; in part by Zhejiang Provincial Natural Science Foundation of China under Grant LQ20F030014 and Grant LY20F020022; and in part by the Royal Society of U.K.

ABSTRACT Preserving an appropriate population diversity is critical for the performance of evolutionary algorithms. In this paper, we present a co-evolutionary niching strategy (CoEN) to dynamically evolve appropriate niching methods and incorporate it into differential evolution (DE) to preserve the population diversity. The proposed CoEN strategy is achieved by optimizing a criterion, which involves both fitness improvement and population diversity resulting from employing the niching methods during evolution of DE. To verify the performance of proposed method, an extensive test on benchmark functions taken from CEC2019 and CEC2014 has been carried out. The results show the significance of proposed CoEN and, by incorporating CoEN, the resulting DE is able to achieve a better or competitive performance than related algorithms.

INDEX TERMS Niching method, evolutionary algorithm, crowding, restricted tournament selection, global optimization.

I. INTRODUCTION

Optimization problems are frequently encountered in many fields such as engineering design, data analysis, financial planning and business [1]. The global optimization algorithms aim to identify the optimal or near-optimal solution. As the solution space usually involves in many local optima, avoiding these local optima is a major challenge for global optimization. Evolutionary algorithms (EAs) [2]–[4], which are able to avoid local optima of solution space, have been widely employed for global optimization [5], [6]. Among various EAs, differential evolution (DE) [7], [8] turns out to be a viable EA. Unlike other EAs, DE generates offspring by employing scaled differences among randomly sampled individuals in the population, which makes it self-adaptive to the fitness landscape of search space [9]. However, like other EAs, DE suffers from premature convergence, especially, for the problem involving in complex search space with many local optima.

To address premature convergence, preserving the population diversity is critical and many schemes have been proposed. Among these schemes, the niching method is perhaps the most popular. The niching method is referred to the technique of preserving multiple niches or favorable parts of the solution space possibly around multiple solutions. Niching methods have been widely incorporated into EAs to identify multiple solutions of the problem [10]. As the niching methods can be used to preserve population diversity, they have also been employed by EAs to avoid local optima of solution space. For example, niching methods including crowding, speciation and fitness sharing have been incorporated into DE to preserve population diversity [11]–[15]. Existing niching based EAs, however, typically employ a single niching method during evolution [16]–[20]. Since different niching methods possess different diversity preserving properties, employing a fixed niching method during DE evolution may have a limited performance. To alleviate this issue, niching ensemble strategies [21], [22], which ensemble multiple niching schemes or employ a certain niching method with various parameter settings have been introduced and incorporated into EAs [14], [23]–[25]. These studies show...
that using multiple niching methods is able to improve the performance of niching based EAs, which rely on a single niching scheme. However, in these strategies, the niching methods are not allowed to dynamically change during the evolution process, thus limiting their performance.

In this paper, a co-evolutionary niching strategy (CoEN), which can dynamically evolve appropriate niching schemes, is proposed and incorporated DE for optimization. The main goal of this study is to produce a flexible, dynamic niching strategy, which can synergistically generate appropriate niching schemes depending on the situation of evolution. This study differs significantly with previous studies, which rely on a single fixed niching scheme or ensemble niching scheme. The proposed strategy has been incorporated into JADE and evaluated on benchmark functions taken from CEC2019 and CEC2014. The results show the significance of proposed CoEN and, by incorporating the CoEN, the resulting DE is able to achieve better or competitive performance than related algorithms.

The rest of the paper is divided into five sections. Related studies are reviewed in Section II. The proposed method is presented in Section III. After that, we report experimental results in Section IV along with an analysis and discussion. Finally, a summary and remarks of the study are given in Section V.

II. RELATED WORKS

This section will first briefly review JADE, which is used as a base for our proposed co-evolutionary niching-based DE algorithms. This is followed by reviewing DE variants, niching-based DE algorithms as well as ensemble and adaptive niching-based algorithms.

DE is a simple yet powerful EA for global optimization [26]. Since the concept of DE was formally introduced by Storn and Price [7], it has attracted an increasing research interest. Many variants of DE have been proposed, among which JADE represents one of state-of-art DE. JADE implements a mutation strategy called “DE/current-to-pbest” with an optional external archive. The DE/current-to-pbest mutation strategy is defined as:

\[ v_{i,g} = x_{i,g} + F_i(x_{\text{best},g} - x_{i,g}) + F_i(x_{r1,g} - x_{r2,g}), \]  

where \( x_{i,g}, x_{r1,g} \) and \( x_{\text{best},g} \) are randomly chosen from one of the top 100% individuals in the current population \( P \) with \( p \in (0, 1) \) while \( x_{r2,g} \) is randomly selected from the union \( P \cup A \). Here, the archive \( A \) is used to collect a set of inferior solutions. In JADE, at the beginning of each generation, the crossover rate (CR) value corresponding to each individual is randomly initialized with a normal distribution of mean \( \mu_{CR} \) and standard deviation of 0.1, while the scale factor (F) is randomly initialized with a Cauchy distribution with a location parameter \( \mu_F \) and standard deviation of 0.1. After each generation, the \( \mu_{CR} \) is updated based on an arithmetic mean (\( \text{mean}_A \)) of successful CR values, while \( \mu_F \) is updated based on a Lehmer mean (\( \text{mean}_L \)) of successful F values. The procedure of JADE is shown in Algorithm 1.

Apart from JADE, many other DE variants have also been proposed. For example, in [27], a Success-History based Adaptive DE (SHADE) is proposed, in which the information of recently successful parameter settings is used to adaptively adjust the parameters of DE. LSHADE [28] extends SHADE algorithm by embedding a linear population size reduction scheme. The performance of LSHADE algorithm is further enhanced by Awad et al. [29], in which the values of F and CR are configured self-adaptively. In [30], Brest et al. devised a DE variant called iLSHADE to improve LSHADE by employing a memory based parameter updating mechanism. In [25], Awad et al. proposed to adaptively set the control parameters by using an ensemble sinusoidal mechanism. While in [31], Meng et al. tried to improve traditional DE by suggesting a grouping strategy for parameter adaptation along with a parabolic population size reduction scheme (PaDE). In this method, a grouping strategy is suggested to properly set the crossover rate during evolution along with a parabolic population size reduction scheme, which is devised to tackle the drawback of linear population size reduction scheme. Additionally, a time stamp based mutation strategy is introduced to avoid inferior solutions as well as to enhance the diversity of external individuals. In [33], Cui et al. proposed to enhance traditional DE based on the concept of working specialization. In this method, the population is divided into sub-populations, which are assigned with different DE strategies for either exploitation or exploration. In [34], a hybrid framework, named HMJCDE, which combines the merits a modified JADE (MJADE) and a modified CoDE (MCoDE), is developed to deal with global optimization problems. The results show HMJCDE is generally able
to outperform state-of-the-art methods in terms of accuracy, convergence rate and robustness. In [35], Cui et al. tended to strengthen the performance of DE by obtaining the guidance of each individual from multiple elites concurrently and independently. Apart from DE, the recently proposed nature-inspired algorithm, named gaining-sharing knowledge based algorithm (GSK) [32], has also shown to be promising for global optimization. The GSK, which is inspired by the process of gaining and sharing knowledge during the human life span, is found to compare favorably to many biologically inspired and nature inspired optimization methods.

Niching methods can assist EAs to preserve the capability to explore the search space by maintaining a diverse population. Thomsen [36] proposed a crowding-based DE to preserve population diversity. Li et al. [37] proposed a fitness sharing scheme based Particle Swarm Optimizer (PSO) for optimization. Gong et al. [38] employed the neighborhood information of individuals to divide a population into niches. In [15], species and crowding-based niching methods were incorporated into DE to deal with multimodal optimization problems. In [39], a structure of hierarchical tree is used to form niches for preserving population diversity during evolution. In this method, nodes located at the top level of the tree denote centers of niches while the nodes beneath each of them represent the individual members of that niche. In [40], a niching DE algorithm, termed as automatic niching DE, was proposed for solving multimodal optimization problems.

Typically, the niching based EAs are based on a single niching scheme. Since different niching methods possess different diversity preserving properties, employing a fixed niching method during EA evolution may have a limited performance. To address this issue, the idea of ensemble [22], [24] has been adopted to employ multiple niching schemes. The existing ensemble niching methods generally differ in the following three aspects: the components of niching candidate pool, the configuration of parameter candidate pool, and the rule of selecting candidate niching schemes. For example, in [23], four niching methods are assigned to four subpopulations and each is responsible to evolve one subpopulation. In [41], an ensemble of restricted tournament selection (RTS) with different window sizes is incorporated into a DE algorithm for global optimization. Similarly, an ensemble of clearing method with different parameter values has also been proposed and embedded into DE in [14] to handle multimodal problems, where the population is divided into three subpopulations and each of them is assigned with a parameterized clearing method. These ensemble strategies could be used to alleviate the issue of using a single fixed niching method to a certain extend. However, the niching schemes used in these strategies are fixed, rather than allowing to dynamically change according to the search situation, thus having a limited performance.

It should also be mentioned that the performance of niching schemes rely on the setting of their parameters. For example, in the fitness sharing scheme, the parameter of niche radius is closely related to its effectiveness. It has been widely established that using a fixed niche radius during evolution may significantly deteriorate its performance. Many attempts have therefore been made to cope with this issue by proposing adaptive parameter control strategies. For example, in [42], Jelasity and Dombi adopted a radius function to control niche radius. While Dick [43] employed the fitness landscape information to identify a single uniform niche radius for all niches. Other works to adaptively set the niche radius can be found in [44]–[46].

### III. CO-EVOLUTIONARY NICHING BASED DE (CoENDE)

The basic idea of co-evolutionary niching strategy lies in the synchronous mutating and updating niching schemes during the process of evolution. In the proposed method, the mutation probability associated with each niching scheme is generated based on a criterion, which considers both fitness improvement and population diversity resulting from its application during evolution. In the following sections, we should describe the details of proposed strategy. The proposed strategy will be incorporated into JADE for evaluation. In JADE, the optional archive $A$ is used to record a set of inferior solutions. Apart from the optional archive $A$, in our method, an external archive $B$ has also been introduced to preserve the elites found at each population.

The procedure of the resulting method, CoENDE, is presented in Algorithm 2.

Algorithm 2: Procedure of CoENDE

**Input:** Define objective function $f(x)$, $x = (x_1, x_2, \ldots, x_d)$

**Output:** The best solutions

Set $\mu = 0.5$, $\mu = 0.5$, $\mu = 30 + \text{ones}(1, NN)$; $\mu = 5 * \text{ones}(1, NN)$

Initialize two archives $A = \emptyset$, $B = \emptyset$

Initialize the sliding window $SW = \emptyset$ and the probability $\text{Prob} = 0$

Find the initial population $\text{POP} = \text{subPop1, subPop2, subPop3}$

Create the initial niche scheme population $\text{Npop}$ using Eq. (2);

for $g = 1$ to $NGs$

for $j = 1$ to $NN$

$S = \text{normrnd}(\mu, 0.5)$

Select niching scheme $N_j$ from $\text{Npop}$

According to $N_j$, generate $\text{mattingPool}$ using Algorithms 3-5

Use Eq. (1) to generate a trial vector

Calculate $\text{div}_1$ and $\text{div}_2$ using Eqs. (7) and (8);

if $\text{div}_1 > \text{div}_2$

$S \leftarrow S$

Update $\text{POP}$ using $N_j$

if $f(x_i) \leq f(x_j)$ and $x_i \notin B$

Remove the first individual from $B$ when $|B| \geq NN$

$B.add(x_i)$

else

$A.add(x_i)$

$S = \text{CR}_S, S = \text{F}_i, S = W$

Update the sliding window $SW \leftarrow \text{min}(f(x_{\text{best}}), S, f_{-1})$

Calculate $\text{Prob}$ and $Q$ according to Eqs. (3) and (4)

Evolve $\text{Npop}$ according to $\text{Prob}$

Update the values of parameters according to Eqs. (9)-(12).
A. NICHING CANDIDATE COMPONENT POOL

Generally, a good niching candidate component pool (NCP) should be able to supply various niching components with diverse characteristics. Here, six niching components are employed in this study, among which three niching components are used in the phase of niching selection, while the other three are used in the phase of niching replacement. Additionally, no niching operators in these two phases have also been considered. Specifically, NCP consists of a niching selection component pool (SCP) and a niching replacement component pool (RCP) as follows:

- SCP:
  - genotype-based randomly selection (GRS)
  - genotype-based greedy selection (GGS)
  - genotype-based probabilistic selection (GPS)
  - no niching selection (denoted as WNS)

- RCP:
  - standard crowding (SC)
  - deterministic crowding (DC)
  - restricted tournament selection (RTS)
  - no niching replacement (denoted as WNR)

The procedure of these niching components are shown in Algorithms 3-8, where \( P_{OP} \) and \( x_i \) represent the population and the current individual, respectively, \( NP \) is a scalar used to control the size of \( P_{OP} \) and \( S \) refers to the sample size.

Based on the NCP, a niching scheme can be simply formed by pairing a niching selection component in SCP with a niching replacement component in RCP along with their associated parameters. As the six niching components possess different degrees of selection or replacement pressure, the formed niching schemes thus have various characters for diversity preservation. Consequently, the NCP can be appropriately used to serve for dynamic niching scheme generation and evolution.

Algorithm 3: The Process of GRS

Input: \( x_i, P_{OP}, NP, S \)
Output: matingPool
\[ D = \text{distance}(x_i, P_{OP}(k)); \]
\[ K = \min(2S, NP); \]
Generate a random set: \( \{x_i \in \{1, NP\} \mid t = (1, 2, \ldots, K)\}; \]
Select \( K \) individuals according to the random set;
Sort these selected individuals according to \( D \) in a descend order;
The top \( S \) out of \( K \) sorted individuals will be added into matingPool.

Algorithm 4: The Process of GGS

Input: \( x_i, P_{OP}, NP, S \)
Output: matingPool
Calculate the distances: \( D = \text{distance}(x_i, P_{OP}); \)
Sort \( P_{OP} \) according to \( D \) in a descend order;
\[ K = \min(2 \times S, NP); \]
Select top \( K \) individuals from \( P_{OP}; \)
The \( S \) best individuals of these selected individuals will be added into matingPool.

Algorithm 5: The Process of GPS

Input: \( x_i, P_{OP}, r \in (0, 1), S \)
Output: matingPool
Initialize \( K = \emptyset, C = \emptyset; \)
\[ D = \text{distance}(x_i, P_{OP}); \]
\[ D_{\text{max}} = \max(D); \]
for \( j = 1:NP \) do
  if \( \text{fitness}(j) \geq r \times \text{fitness}(r) \) then
    matingPool.add\((P_{OP}(j)); C++;
Randomly select \((S - C)\) individuals from pop, and add them into matingPool.

Algorithm 6: The Process of SC

Input: \( P_{OP}, Children, NP \)
Output: \( P_{OP} \)
for \( j = 1:NP \) do
  Calculate the distance values of the \( j^{th} \) child in Children to the other individuals in \( P_{OP}; \)
  Compare the fitness value of the \( j^{th} \) child and the fitness value of the individual that has the smallest Euclidean distance. The child will replace the individual if it is fitter than the individual.
  Stop if the termination criterion is met, otherwise repeat the loop above.

Algorithm 7: The Process of DC

Input: \( P_{OP}, NP \)
Output: \( P_{OP} \)
for \( j = 1:2:NP \) do
  Randomly select two parents \( P_1 \) and \( P_2 \) with replacement (i.e., the selected individuals will not be removed from the population) from the current population;
  Generate two offspring \( C_1 \) and \( C_2 \) using the variation operators;
  Evaluate the fitness value of offspring, and calculate their distance to parents;
  Identify a close competition pair;
  if \( [d(P_0, C_0) + d(P_1, C_1)] \leq [d(P_0, C_1) + d(P_1, C_0)] \) then
    the competition is between \( P_0 \leftrightarrow C_0 \) and \( P_1 \leftrightarrow C_1; \)
  else
    the competition is between \( P_0 \leftrightarrow C_1 \) and \( P_1 \leftrightarrow C_0; \)
  Determine the winner. Individuals with higher fitness values win the competition and remain in the population.Losers are discarded.

Algorithm 8: The Process of RTS

Input: \( P_{OP}, Children, NP, W \)
Output: \( P_{OP} \)
for \( j = 1:2:NP \) do
  Calculate the fitness values of offspring \( f(c1) \) and \( f(c2). \)
  Randomly select \( W \) individuals from the current population with replacement (i.e., the selected individuals will not be removed from the population) as the comparison set;
  Select two closest individuals \( f(d1) \) and \( f(d2) \) in the comparison set to \( f(c1) \) and \( f(c2); \)
  if \( f(c1) > f(d1) \) then
    Replace \( d1 \) with \( c1; \)
  else
    Keep \( d1 \) and discard \( c1; \)
The same operation is carried out between \( d2 \) with \( c2. \)
The goodness of a niching scheme will be evaluated and the results will be used to evaluate niching schemes according to the current search situation. The niching evolution aims to dynamically evolve appropriate niching schemes for evolving niching schemes. The interface layer is responsible for niching scheme generation, while the optimization layer contains a niching-based DE for evolution.

The workflow of CoENDE is shown in Fig. 2. In CoENDE, the evolution is divided into multiple evolution periods (i.e., the learning period, $L_P$). At each generation, an offspring population will be generated from the parent population. During each generation, certain well-performed pivotal parameter values will be memorized in predefined memories. At the end of each generation, the control parameters of each individual $x_i$ will be independently generated according to certain adaptive rules. At the end of $L_P$ generations, the goodness of niching schemes will be evaluated and the results will be used to evolve the niching schemes, details of which are presented in the following subsections.

C. NICHING EVOLUTION

The niching evolution aims to dynamically evolve appropriate niching schemes according to the current search situation. Specifically, the process of niching evolution works as follows. First, a niching scheme population (NPop) is initialized by randomly selecting one component from selection and replacement component pools, respectively, as:

$$NPop = \{N_1, \ldots, N_m\}, \quad (2)$$

where $m$ is the size of NPop, $N_k$ ($k = 1, \ldots, m$) is the $k$th niching scheme. Then, all niching schemes will be employed at least once to evolve the population. After that, all niching schemes will undergo a mutation operation, the probability of which is calculated as:

$$P_k = 1 - Q_k / \max(Q), \quad (3)$$

where $P_k$ and $Q_k$ denote the mutation probability and the goodness of $k$th niching scheme, respectively. $Q$ is a set of goodness evaluations of all niching schemes and $\max(\cdot)$ denotes the maximum value. Based on the above equation, a niching scheme suitable to the current evolution stage will have a small probability to be mutated. Otherwise, a high mutation probability will be assigned. The mutation operation works by randomly changing the selection and replacement component of the niching scheme.

To evaluate the goodness of a niching scheme, both the fitness improvement (FI) and population entropy (PE) have been considered. To be specific, we calculate the goodness of a niching scheme based on a weighted sum of the ranking score of FI and PE as:

$$Q_k = \omega \cdot \text{score}(FI_k) + (1 - \omega) \cdot \text{score}(PE_k), \quad (4)$$

where $\text{score}(\cdot)$ means a ranking score and $\omega \in (0, 1)$ is a weight factor. $PE$, which refers to the entropy of current population, is calculated as:

$$PE_k = -\sum_{i=1}^{np} p_i \ast \log(p_i), \quad (5)$$

where $p_i$ is the probability of ith non-repeating individual, $np$ refers to the number of different individuals and $\log(*)$ is a logarithm function. To calculate $FI$, the best fitness values achieved so far are first stored in a queue memory (i.e., sliding window). Fig. 3 shows the structure of a queue memory. Then, $FI$ is defined as the sum of difference between the best fitness value and each solution in the queue as:

$$FI_k = \sum_{i=1}^{W} (f_{\text{best}} - SW(i)), \quad (6)$$

where $W$ represents the length of sliding window (denoted by $SW$), $SW(i)$ denotes the fitness of $i$th solution stored in $SW$ and $f_{\text{best}}$ is the best fitness value in the current generation. During evolution, if the number of individuals exceeds the predefined value of $W$, the earliest individuals will be removed.

D. PARAMETER ADAPTATION

The niching scheme generated with the six niching components involves certain parameters, which need to be properly set. For niching selection components, here we introduce a distance-based self-adaptive strategy to set their corresponding parameter values. Specifically, for each selection component, the corresponding parameter value is first randomly initialized with a normal distribution of mean $\mu_s$ and standard deviation 0.5. At the beginning of each generation, the disparity between the current population and trial vectors is then calculated as:

$$\text{div}_1(i) = \sum_{j=1}^{NP} (\text{dist}(x_i, x_j)), \quad (7)$$

$$\text{div}_2(i) = \sum_{j=1}^{NP} (\text{dist}(x_i, u_j)), \quad (8)$$

where $\text{dist}(\cdot)$ denotes the Euclidean distance, $\text{div}_1$ and $\text{div}_2$ indicate the population diversity, $x_i$ and $x_j$ are the $i$th and $j$th
individual of population, and $u_j$ represents the $j$th trial vector. The current parameter setting is thought to be successful if $\text{div}_2$ is larger than $\text{div}_1$. In this case, the current parameter value will be saved in a variable $S_S$, and the parameter value is updated using an arithmetic mean of the values in $S_S$. For the replacement component, successful parameter values will be first recorded in a variable $S_R$. Then, a Lehmer mean of $S_R$ will be used to update these parameters at the end of each generation. The updating rules for parameters of niching schemes are defined as:

$$\mu_{S_j} = (1 - c) \cdot \mu_{S_j} + c \cdot \text{mean}_A(S_S)$$

$$\mu_{R_j} = (1 - c) \cdot \mu_{R_j} + c \cdot \text{mean}_L(S_R)$$

For the crossover probability and mutation factor in DE, they are updated according to the scheme proposed in JADE. Specifically, at each generation $g$, the crossover probability $CR_i$ and mutation factor $F_i$ of each individual $x_i$ are calculated independently based on a normal distribution of mean $\mu_{CR}$ with standard deviation 0.1 and a Cauchy distribution with location parameter $\mu_F$ and scale parameter 0.1, respectively. At the end of each generation, $\mu_{CR}$ and $\mu_F$ are updated as:

$$\mu_{CR_j} = (1 - c) \cdot \mu_{CR_j} + c \cdot \text{mean}_A(S_{CR})$$

$$\mu_{F_j} = (1 - c) \cdot \mu_{F_j} + c \cdot \text{mean}_L(S_{F})$$

where $\mu_{CR_j}$ and $\mu_{F_j}$ are initialized to be 0.5.

**IV. EXPERIMENTS**

**A. DATA SETS AND PARAMETER SETTING**

Twenty-four benchmark functions taken from [47] and [48] have been used to evaluate the performance of CoENDE. The functions to be tested involve multimodal functions with shift and rotation, hybrid functions and composition functions.

| No. | Functions                                           | Dimension | Search Range        | $F_{\text{min}}$ | $\text{NGs}$ |
|-----|-----------------------------------------------------|-----------|---------------------|------------------|--------------|
| 1   | Storm’s Chebyshev Polynomial Fitting Problem        | 9         | $[-8192,8192]$     | 1                | 5000         |
| 2   | Inverse Hilbert Matrix Problem                      | 16        | $[-16384,16384]$   | 1                | 20000        |
| 3   | Leonard-Jones Minimum Energy Cluster Problem       | 18        | $[-4,4]$           | 1                | 10000        |
| 4   | Shifted and Rotated Rastrigin’s Function            | 10        | $[-100,100]$       | 1                | 10000        |
| 5   | Shifted and Rotated Griewank’s Function             | 10        | $[-100,100]$       | 1                | 5000         |
| 6   | Shifted and Rotated Weierstrass Function            | 10        | $[-100,100]$       | 1                | 5000         |
| 7   | Shifted and Rotated Schwefel’s Function             | 10        | $[-100,100]$       | 1                | 5000         |
| 8   | Shifted and Rotated Expanded Schaffer’s PS Function | 10        | $[-100,100]$       | 1                | 20000        |
| 9   | Shifted and Rotated Happy Cat Function              | 10        | $[-100,100]$       | 1                | 5000         |
| 10  | Shifted and Rotated Ackley Function                 | 10        | $[-100,100]$       | 1                | 10000        |
| 11  | Hybrid Function 1                                   | 10, 30, 50| $[-100,100]$       | 1700             | 20000        |
| 12  | Hybrid Function 2                                   | 10, 30, 50| $[-100,100]$       | 1400             | 20000        |
| 13  | Hybrid Function 3                                   | 10, 30, 50| $[-100,100]$       | 1900             | 20000        |
| 14  | Hybrid Function 4                                   | 10, 30, 50| $[-100,100]$       | 2000             | 20000        |
| 15  | Hybrid Function 5                                   | 10, 30, 50| $[-100,100]$       | 2100             | 20000        |
| 16  | Hybrid Function 6                                   | 10, 30, 50| $[-100,100]$       | 2200             | 20000        |
| 17  | Composition Function 1                              | 10, 30, 50| $[-100,100]$       | 2300             | 20000        |
| 18  | Composition Function 2                              | 10, 30, 50| $[-100,100]$       | 2400             | 20000        |
| 19  | Composition Function 3                              | 10, 30, 50| $[-100,100]$       | 2500             | 20000        |
| 20  | Composition Function 4                              | 10, 30, 50| $[-100,100]$       | 2600             | 20000        |
| 21  | Composition Function 5                              | 10, 30, 50| $[-100,100]$       | 2700             | 20000        |
| 22  | Composition Function 6                              | 10, 30, 50| $[-100,100]$       | 2800             | 20000        |
| 23  | Composition Function 7                              | 10, 30, 50| $[-100,100]$       | 2900             | 20000        |
| 24  | Composition Function 8                              | 10, 30, 50| $[-100,100]$       | 3000             | 20000        |

* $f_1 - f_{10}$ are taken from CEC2019, $f_{11} - f_{24}$ are taken from CEC2014.

**B. COMPARED ALGORITHMS**

The proposed CoENDE will be compared with the following algorithms:

The test functions possess various dimensions. For example, functions $f_1$, $f_2$ and $f_3$ have 9, 16, and 18 dimensions, respectively, while $f_4$-$f_{10}$ carry a dimension of 10. To ensure fair comparisons, the same set of initial populations are used for all methods and the results are averaged over 50 trials of each method. The information of these functions are shown in Table 1, where $D$ denotes the dimensionality of the function, $F_{\text{min}}$ is the optimal value of the function and $\text{NGs}$ denotes the number of maximum generations. The settings of parameters of the proposed method are listed in Table 2.

In the proposed method, the purpose of introducing $L_P$ is to make sure that the algorithm could run for a period of time in the current state to acquire more information from the optimization process. While, the $SW$ is a queue memory, which is used to dynamically store the best fitness values achieved so far by each niching method. The value of $L_P$ and the size of $SW$ are set experimentally. Specifically, ten trials of the algorithm with various values of $L_P$ and $SW$ sizes have been tested. The one, which delivers the best average fitness, has been chosen to set the value of $L_P$ and the size of $SW$.  

| Parameters | Values | Descriptions |
|------------|--------|--------------|
| $NP$       | 100    | Size of each subpopulation |
| $NN$       | 3      | Size of niching scheme population |
| $F$        | 0.5    | Amplification factor of DE |
| $CR$       | 0.9    | Crossover rate of DE |
| $\omega$   | 0.2    | Weight factor of Eq (4) |
| $L_P$      | 400    | Learning period |
| $W$        | 100    | Length of the sliding window |
### Table 3: Comparing the performance of CoENDE, JADE, PaDE, GSK, SHADE, LSHADE, LSHADE-EpSin, ENA, EPSO, NSDE, CrowdingDE, DCDE, RTSDE, DCGRSDE and RTSGRSDE on benchmark functions $f_1$–$f_{10}$.

| Algorithm | Mean (Std dev) | Mean (Std dev) | Mean (Std dev) | Mean (Std dev) | Mean (Std dev) | Mean (Std dev) |
|-----------|----------------|----------------|----------------|----------------|----------------|----------------|
| CoENDE    | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       |
| JADE      | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       |
| PaDE      | 5.79E-02       | 2.36E-02       | 2.34E-02       | 2.39E-02       | 2.35E-02       | 2.37E-02       |
| GSK       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       |
| SHADE     | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       | 0.00E+00       |
| LSHADE    | 9.26E-02       | 2.09E-02       | 2.39E-02       | 2.08E-02       | 2.12E-02       | 2.10E-02       |
| LSHADE-Ep | 8.30E-02       | 1.20E-02       | 1.16E-02       | 1.18E-02       | 1.15E-02       | 1.17E-02       |
| ENA       | 5.23E-02       | 1.18E-02       | 1.17E-02       | 1.18E-02       | 1.16E-02       | 1.18E-02       |
| EPSO      | 5.36E-02       | 1.19E-02       | 1.17E-02       | 1.18E-02       | 1.16E-02       | 1.18E-02       |
| NSDE      | 5.78E-02       | 1.19E-02       | 1.17E-02       | 1.18E-02       | 1.16E-02       | 1.18E-02       |
| CrowdingDE| 5.52E-02       | 1.19E-02       | 1.17E-02       | 1.18E-02       | 1.16E-02       | 1.18E-02       |
| DCGRSDE   | 5.60E-02       | 1.19E-02       | 1.17E-02       | 1.18E-02       | 1.16E-02       | 1.18E-02       |
| RTSGRSDE  | 5.62E-02       | 1.19E-02       | 1.17E-02       | 1.18E-02       | 1.16E-02       | 1.18E-02       |

### Table 4: Comparing the performance of CoENDE, JADE, PaDE, GSK, SHADE, LSHADE, LSHADE-EpSin, ENA, EPSO, NSDE, CrowdingDE, DCDE, RTSDE, DCGRSDE and RTSGRSDE on 10-dimensional benchmark functions $f_1$–$f_{10}$.

| Algorithm | Mean (Std dev) | Mean (Std dev) | Mean (Std dev) | Mean (Std dev) | Mean (Std dev) | Mean (Std dev) |
|-----------|----------------|----------------|----------------|----------------|----------------|----------------|
| CoENDE    | 3.40E-01       | 4.87E-03       | 1.10E-02       | 7.83E-03       | 6.81E-03       | 7.83E-03       |
| JADE      | 5.04E-01       | 1.41E-03       | 1.54E-03       | 1.41E-03       | 1.54E-03       | 1.41E-03       |
| PaDE      | 6.34E-01       | 2.17E-03       | 2.31E-03       | 2.17E-03       | 2.31E-03       | 2.17E-03       |
| GSK       | 3.85E-01       | 1.14E-03       | 1.14E-03       | 1.14E-03       | 1.14E-03       | 1.14E-03       |
| SHADE     | 8.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |
| LSHADE    | 9.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |
| LSHADE-Ep | 9.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |
| ENA       | 9.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |
| EPSO      | 9.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |
| NSDE      | 9.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |
| CrowdingDE| 9.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |
| DCGRSDE   | 9.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |
| RTSGRSDE  | 9.54E-01       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       | 2.00E-03       |

A1. JADE [49]: An adaptive DE with optional external archive.
A2. PaDE [31]: An enhanced DE algorithm with novel control parameter adaptation schemes.
A3. GSK [32]: A gaining-sharing knowledge based algorithm.
A4. SHADE [27]: The success history based adaptive DE.
### TABLE 5. Comparing the performance of CoENDE, JADE, PaDE, GSK, SHADE, LSHADE, LSHADE-EpSin, ENA, EPSO, NShDE, CrowdingDE, DCDE, RTSDE, DCGRSD and RTSGRSDE on 30-dimensional benchmark functions $f_{11} – f_{24}$.

| Algorithm        | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ | $f_{16}$ | $f_{17}$ | $f_{18}$ | $f_{19}$ | $f_{20}$ | $f_{21}$ | $f_{22}$ | $f_{23}$ | $f_{24}$ |
|------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| CoENDE           | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| JADE             | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| PaDE             | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| GSK              | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| LSHADE           | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| LSHADE-EpSin     | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| ENA              | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| EPSO             | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| NShDE            | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| CrowdingDE       | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| DCDE             | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| RTSDE            | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| DCGRSD           | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |
| RTSGRSDE         | 3.94(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  | 3.86(4)  | 9.76(8)  |

A5. LSHADE [28]: A variant of SHADE with linear population size reduction.

A6. LSHADE-EpSin [29]: An ensemble sinusoidal approach based LSHADE.

A7. ENA [23]: A niching ensemble based EA.

A8. EPSO [50]: An ensemble particle swarm optimizer.

A9. NShDE [51]: A neighborhood based sharing DE.

A10. CrowdingDE: Crowding based DE.
A11. DCDE: Deterministic crowding based DE.
A12. RTSDE: Restricted tournament selection based DE.
A13. DCGRSDE: An DE algorithm incorporated with DC and GRS.
A14. RTSGRSDE: An DE algorithm incorporated with RTS and GRS.

In the above algorithms, A1-A9 are classical or recently proposed algorithms, while A10-A14 are variants of the proposed CoENDE, in which one or two niching schemes are employed. All experiments are implemented using MATLAB on a machine running operation system of Ubuntu (version 16.04) with Intel Core i7-8700 and RAM of 16 GB DDR4. To make the comparison fair, all algorithms are implemented using the same population sizes and maximum number of generations. Unless otherwise stated, we report the mean and standard deviation of the difference between obtained best fitness and known optimal values as well as the Wilcoxon rank-sum test at a significance threshold of $p < 0.1$. It should also be noted that the difference is considered to be 0 when it is equal or smaller than $1E10^{-8}$, which is a guideline of CEC2014 test suit.

C. COMPARING CoENDE WITH NON-ENSEMBLE BASED RELATED ALGORITHMS

In this section, we report the results of comparing our proposed method with JADE, JADE based algorithms including SHADE, LSHADE and LSHADE-EpSin as well as two recently proposed algorithms including PaDE and GSK. The results are shown in the top rows of Tables 3-6. A summary of Wilcoxon rank-sum tests of the results between our method and related methods is given in Table 7. The results show CoENDE is generally able to deliver better or comparable performance than the six related algorithms. Specifically, on the functions $f_1$-$f_{10}$, the performance of our method is better than JADE, SHADE, LSHADE-EpSin, GSK and PaDE while comparable with LSHADE. On the hybrid and composition functions $f_{11}$-$f_{24}$, our method could achieve better results than JADE, SHADE, LSHADE and LSHADE-EpSin, comparable performance than GSK while being slightly outperformed by PaDE. Based on the results, clearly, our proposed method is able to appropriate search the space, thus delivering a competitive performance.

D. COMPARING CoENDE WITH ENSEMBLE AND HYBRID EA ALGORITHMS

This section reports the results of comparing CoENDE with two ensemble niching based EAs (i.e., ENA [23] and EPSO [50]) and a hybrid niching based DE (NShDE [51]). In ENA, several existing niching methods are combined into a unified GA framework for optimization. EPSO, on the other hand, focuses on the ensemble of different PSO variants for optimization. NShDE is a sharing-based hybrid niching DE algorithm, in which a neighborhood based mutation is also proposed to generate offspring. The results are reported in middle rows of Tables 3-6. Table 7 shows the results of significance tests.

From the results in Tables 3-6, we can see that CoENDE generally performs better than ENA, EPSO and NShDE on most functions in terms of the mean value. Among the three algorithms to be compared, we can find that NShDE could have the best performance, which is followed by EPSO and ENA. The results of significance tests show that the better performance of CoENDE comparing to ENA, EPSO and NShDE is generally statistically significant. The worse performance of ENA and EPSO is mainly due to they employ a fixed niching pool, which limits their performance. It should be noted that our method could be outperformed by NShDE on a few functions to be tested. However, in most of the cases, the better performance of NShDE is not statistically significant.

E. COMPARING CoENDE WITH ITS VARIANTS

To examine the effectiveness of proposed CoEN strategy, we compare CoENDE with its variants: rather than the CoEN strategy, only a fixed one or two niching schemes is incorporated into the proposed framework for optimization. Five variants (i.e., CrowdingDE, DCDE, RTSDE, DCGRSDE, and RTSGRSDE), in which standard crowding, DC, RTS, GRS and their combinations are employed as the niching scheme, have been compared. The results are reported in last rows of Tables 3-6. The results show that the two RTS-based variants perform relatively better than the rest three variants. However, all variants are outperformed by CoENDE. From Table 7,
the results show that the better performance of our method than its variants is generally statistically significant. This may reveal that CoEN can overcome the drawback of employing fixed niching schemes.

**V. CONCLUSION**

Niching technique has been widely investigated in EA community and existing studies generally focus on employing a single fixed niching scheme or several niching schemes in an ensemble manner. To effectively solve various optimization problems, different niching schemes may be required for an EA to work properly. Even for a specific optimization problem, the most suitable niching scheme may be different at different stages of the evolutionary process. In this paper, we propose to co-evolve the niching schemes, thus allowing appropriate niching schemes to be dynamically generated and employed during DE evolution. The performance of the proposed method has been evaluated on the benchmark functions from CEC2019 and CEC2014. The results show the significance of proposed evolutionary niching scheme and the resulting method could achieve better or comparable performance than related methods.

The proposed method can be extended in a few aspects. Firstly, it is desirable to employ more niching components to evaluate the effectiveness of the proposed niching evolution scheme. Secondly, as the evolved niching schemes are able to preserve multiple niches in the population, the proposed method can be suitably modified to address multimodal optimization problems. Finally, it is interesting to incorporate the devised niching evolution strategy into other meta-heuristics, such as PSO, for optimization.

**REFERENCES**

[1] J. Sun, Q. Zhang, and E. P. K. Tsang, “DE/EDA: A new evolutionary algorithm for global optimization,” *Inf. Sci.*, vol. 169, nos. 3–4, pp. 249–262, 2005.

[2] D. E. Goldberg, “Genetic algorithms in search, optimization, and machine learning,” in *Proc. Ethnobiographic Praxis Ind. Conf.*, 1988, vol. 9, no. 2, pp. 95–99.

[3] R. K. Ursem, “Multinational evolutionary algorithms,” in *Proc. Congr. Evol. Comput. (CEC)*, Jul. 1999, pp. 1633–1640.

[4] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. London, U.K.: Springer-Verlag, 2010.

[5] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. D. Carvalho, “A survey of evolutionary algorithms for clustering,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 2, pp. 133–155, Mar. 2009.

[6] T. Bäck and H.-P. Schwefel, “An overview of evolutionary algorithms for parameter optimization,” *Evol. Comput.*, vol. 1, pp. 1–23, Dec. 1993.

[7] 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. 345–359, 1997.

[8] K. V. Price, “An introduction to differential evolution,” in *New Ideas in Optimization*, D. Corne, D. Marco, and F. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 78–108.

[9] 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.

[10] 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.

[11] R. Mukherjee, G. R. Patra, R. Kundu, and S. Das, “Cluster-based differential evolution with crowding archive for niching in dynamic environments,” *Inf. Sci.*, vol. 267, pp. 58–82, May 2014.

[12] D. J. Poole and C. B. Allen, “Constrained niching using differential evolution,” *Swarm Evol. Comput.*, vol. 44, pp. 74–100, Feb. 2019.

[13] 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.

[14] B. Qu, J. Liang, P. N. Suganthan, and T. Chen, “Ensemble of clearing differential evolution for multi-modal optimization,” in *Proc. Int. Conf. Adv. Swarm Intell.*, vol. 7331, Jun. 2012, pp. 350–357.

[15] S. Biswas, S. Kundu, and S. Das, “Inducing niching behavior in differential evolution through local information sharing,” *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 124–263, Apr. 2015.

[16] G. Y. Yang, Z. Y. Dong, and K. P. Wong, “A modified differential evolution algorithm with fitness sharing for power system planning,” *IEEE Trans. Power Syst.*, vol. 23, no. 2, pp. 514–522, May 2008.

[17] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, “Finding multiple global optima exploiting differential evolution’s niching capability,” in *Proc. IEEE Symp. Diff. Evol. (SDE)*, Apr. 2011, pp. 1–8.

[18] J. Montgomery and S. Chen, “A simple strategy for maintaining diversity and reducing crowding in differential evolution,” in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.

[19] S. Hui and P. N. Suganthan, “Ensemble differential evolution with dynamic subpopulations and adaptive clearing for solving dynamic optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.

[20] S. Hui and P. N. Suganthan, “Ensemble crowding differential evolution with neighborhood mutation for multimodal optimization,” in *Proc. IEEE Symp. Diff. Evol. (SDE)*, Apr. 2013, pp. 135–142.

[21] T. G. Dietterich, “Ensemble methods in machine learning,” in *Multiple Classifier Systems (Lecture Notes in Computer Science)*, vol. 1857. Berlin, Germany: Springer, 2000, pp. 1–15.

[22] S. Zhou and Z. Sun, “Can ensemble method convert a ‘weak’ evolutionary algorithm to a ‘strong’ one?” in *Proc. Int. Conf. Evol. Model. Control Autom. Int. Conf. Web Intell. Agents, Web Technol. Internet Commerce (CIMCA-IWATIC)*, 2005, pp. 68–74.

[23] E. L. Yu and P. N. Suganthan, “Ensemble of niching algorithms,” *Inf. Sci.*, vol. 180, no. 15, pp. 2815–2833, Aug. 2010.

[24] B. Y. Qu, P. Gouthaman, and P. N. Suganthan, “Dynamic grouping crowding differential evolution with ensemble of parameters for multimodal optimization,” in *Proc. Int. Conf. Swarm. Evol., Memetic Comput.*, Dec. 2010, pp. 19–28.

[25] N. H. Awad, M. Z. Ali, and P. N. Suganthan, “Ensemble of parameters in a sinusoidal differential evolution with niching-based population reduction,” *Swarm Evol. Comput.*, vol. 39, pp. 141–156, Apr. 2018.

[26] D. Zaharie, “On the explorative power of differential evolution algorithms,” *Verhandlungen Der Deutschen Gesellschaft Fur Innere Medizin*, vol. 39, pp. 551–554, Oct. 2001.

[27] R. Tanabe and A. Fukunaga, “Success-history based parameter adaptation for differential evolution,” in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 71–78.

[28] R. Tanabe and A. S. Fukunaga, “Improving the search performance of SHADE using linear population size reduction,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665.

[29] N. H. Awad, M. Z. Ali, P. N. Suganthan, and R. G. Reynolds, “An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems,” in *Proc. IEEE Congr. Evol. Comput.* (CEC), Jul. 2016, pp. 2958–2965.

[30] J. Brest, M. S. Maucec, and B. Boskovic, “IL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2014, pp. 1188–1195.

[31] Z. Meng, J.-S. Pan, and K.-K. Tseng, “PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization,” *Knowl.-Based Syst.*, vol. 168, pp. 80–99, Mar. 2019.

[32] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, “Gaining-sharing knowledge based algorithm for solving optimization problems: A novel nature-inspired algorithm,” *Int. J. Mach. Learn. Cybern.*, vol. 11, pp. 1501–1529, Jul. 2020.

[33] L. Cui, G. Li, Q. Lin, J. Chen, and N. Lu, “Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations,” *Comput. Oper. Res.*, vol. 67, pp. 155–173, Mar. 2016.
N. Lynn and P. N. Suganthan, “Ensemble particle swarm optimizer,” J. Zhang and A. C. Sanderson, “JADE: Adaptive multiple-elites-guided composite differential evolution algorithm with a shift mechanism,” Inf. Sci., vol. 422, pp. 122–143, Jan. 2018.

R. Thom森, “Multimodal optimization using crowding-based differential evolution,” in Proc. Congr. Evol. Comput., Jun. 2004, pp. 1382–1389.

T. Li, C. Wei, and W. Pei, “PSO with sharing for multimodal function optimization,” in Proc. Int. Conf. Neural Netw. Signal Process., Dec. 2003, pp. 450–453.

Y.-J. Gong, J. Zhang, and Y. Zhou, “Learning multimodal parameters: A bare-bones niching differential evolution approach,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 7, pp. 2944–2959, Jul. 2018.

S. Kalra, S. Rahnamayan, and K. Deb, “Enhancing Clearing-based niching method using Delaunay triangulation,” in Proc. Congr. Evol. Comput. (CEC), Jun. 2017, pp. 2328–2337.

Z.-J. Wang, Z.-H. Zhan, Y. Lin, W.-J. Yu, H. Wang, S. Kwong, and J. Zhang, “Automatic niching differential evolution with contour prediction approach for multimodal optimization problems,” IEEE Trans. Evol. Comput., vol. 24, no. 1, pp. 114–128, Feb. 2020.

B.-Y. Qu and P. N. Suganthan, “Novel multimodal problems and differential evolution with ensemble of restricted tournament selection,” in Proc. Congr. Evol. Comput., Jul. 2010, pp. 1–7.

J. J. Liang, B. Y. Qu, and P. N. Suganthan, “Problem definitions and evaluation criteria for the 100-digit challenge special session,” in Proc. IEEE Conf. Cybern. Intell. Syst., Dec. 2004, pp. 361–366.

J. Gan and K. Warwick, “Dynamic niche clustering: A fuzzy variable radius niching technique for multimodal optimisation in gas,” in Proc. Congr. Evol. Comput., May 2001, pp. 215–222.

O. M. Shir, “Niching in derandomized evolution strategies and its applications in quantum control,” Ph.D. dissertation, Dept. Comput. Sci., Natural Comput. Group, LIACS, Fac. Sci., Leiden Univ., Leiden, The Netherlands, 2008.

K. V. Price, N. H. Awad, M. Z. Ali, and P. N. Suganthan, “Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization,” Nanyang Technol. Univ., Singapore, Tech. Rep., 2018.

J. J. Liang, B. Y. Qu, and P. N. Suganthan, “Problem definitions and evaluation criteria for the CEC2014 special session and competition on single objective real-parameter numerical optimization,” Zhengzhou Univ., Nanyang Technol. Univ., Henan, China, Tech. Rep., 2013.

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.

N. Lynn and P. N. Suganthan, “Ensemble particle swarm optimizer,” Appl. Soft Comput., vol. 55, pp. 533–548, Jun. 2017.

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.

LE YAN received the M.Sc. degree in control theory and control engineering from the University of Shanghai for Science and Technology, Shanghai, China, in 2017. He is currently pursuing the Ph.D. degree in control science and engineering. His research interests include evolutionary computation, data clustering, neural networks, and set-membership filtering.

JIANJUN CHEN received the M.A. degree from Zhejiang Sci-Tech University, in 2009. She is currently an Associate Professor with Keyi College of Zhejiang Sci-Tech University. Her research interests include dynamic visual design, evolutionary design, and VR imaging.

QI LI received the B.Eng. degree in electrical engineering and automation from Jiangsu University of Technology, Changzhou, China, in 2013, and the Ph.D. degree in control science and engineering from Donghua University, Shanghai, China, in 2018.

From June 2016 to July 2016, she was a Research Assistant with the Department of Mathematics, Texas A&M University at Qatar, Qatar. From November 2016 to November 2017, she was a Visiting Ph.D. Student with the Department of Computer Science, Brunel University London, U.K. She is currently a Lecturer with Hangzhou Normal University, Hangzhou, China. Her current research interests include network communication, complex networks, and sensor networks. She is a very active reviewer for many international journals.

JIAFA MAO received the Ph.D. degree in pattern recognition from the East China University of Science and Technology, China, in 2009. Since then, he has been working as a Postdoctoral Researcher at Beijing University of Posts and Telecommunications, China. In July 2011, he joined Zhejiang University of Technology, China, where he is currently a Professor with the School of Computer Science and Technology. He has published over 50 papers in the scientific literature. His research interests include pattern recognition, digital image processing, and information hiding.

WEIGUO SHENG (Member, IEEE) received the M.Sc. degree in information technology from the University of Nottingham, U.K., in 2002, and the Ph.D. degree in computer science from Brunel University London, U.K., in 2005. Then, he worked as a Researcher at the University of Kent, U.K., and Royal Holloway, University of London, U.K. He is currently a Professor with Hangzhou Normal University. His research interests include evolutionary computation, data mining/clustering, pattern recognition, and machine learning.