Neurocomputing 396 (2020) 24–38

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Concurrent optimization of multiple base learners in neural network ensembles: An adaptive niching differential evolution approach☆

Ting Huang[a,1], Dan-Ting Duan[b,1], Yue-Jiao Gong[a,*], Long Ye[c], Wing W.Y. Ng[a], Jun Zhang[d]

[a] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[b] Key Laboratory of Media Audio and Video of Ministry of Education, Communication University of China, Beijing, China
[c] State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing, China
[d] University of Victoria, Melbourne, VIC, Australia

ARTICLE INFO

Article history:
Received 27 August 2019
Revised 20 January 2020
Accepted 3 February 2020
Available online 7 February 2020

Communicated by Dr Weiguo Sheng

Keywords:
Niching differential evolution
Multimodal optimization
Neural network ensemble
Population size adaptation

ABSTRACT

Neural network ensemble (NNE) exhibits improved performance when compared with a single neural network (NN) in most cases. Traditionally, each base network in an NNE is trained individually, which may result in network redundancy and expensive training overhead. This paper proposes a new adaptive niching evolutionary algorithm, which possesses promising performance in finding multiple optima in terms of good accuracy and diversity. By means of this algorithm, all NNs in an NNE can be trained simultaneously. In particular, the proposed algorithm is named adaptive niching differential evolution (ANDE), which is characterized by a heuristic clustering method to enable iteratively cluster subpopulations that track and locate multiple optima, a parameter adaptation strategy to adaptively adjust parameters according to the subpopulation states, and an auxiliary movement scheme to promote the equilibrium between exploration and exploitation. Experimental results validate the efficiency and effectiveness of the proposed ANDE on the benchmark test suite of multimodal optimization. Furthermore, ANDE is extended to concurrently train multiple base NNs for ensemble and the experiments show a promising performance of ANDE-NNE.

1. Introduction

As one of the most common methods in machine learning, neural networks (NNs) have been extensively researched and employed in various fields [1]. Subsequently, neural network ensemble (NNE), proposed by Hansen and Salamon [2], improves the generalization performance of NNs by combining the training results of multiple NNs. In general, NNE is more competitive than a single NN. Recently, NNE has found wide applications in wide real-world fields, including data mining [3], medical diagnosis [4], protein-protein interactions [5], face recognition [6], energy forecasting [7], taffic scheduling[8], and crime prevention [9,10].

The performance of NNE greatly depends on the accuracy and diversity of the base NNs [11,12]. Evolutionary algorithms (EAs) [13–20], a branch of population-based metaheuristic algorithms, have exhibited promising performance in the searching accuracy and diversity. Considering this, the EAs provide a potential way to train the NNE effectively. There are several studies in the related area. Kwon et al. [21] proposed a parallel genetic algorithm to optimize the weight values of NNs. In [22], three different EAs were adopted to optimize three different NNs for ensemble. Sheng et al. [23] utilized a fitness sharing method to preserve the population diversity and designed a penalty coefficient to balance the diversity and accuracy. However, there remain some open questions in this area, such as how to differentiate the training of different base NNs and how to control the number of NNs to ensemble. Meanwhile, the previous studies empirically set the parameters of EAs for different problem instances, which may not be efficient all the time.

In this study, we model the training of NNE as a multimodal optimization problem (MMOP). MMOP has multiple global optima in the problem space and requires the optimizer to obtain multiple high-quality and diverse solutions simultaneously. In the MMOP-based NNE training, each optimal solution of MMOP corresponds

☆ This work was supported in part by the National Natural Science Foundation of China under Grants. 61873095, 61873097, 61971383, and U1701267, in part by the Guangdong Natural Science Foundation Research Team Project under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform Project under Grant 2018B050502006.

* Corresponding author.
E-mail address: gongyuejiao@gmail.com (Y.-J. Gong).

[1] The authors contributed equally to this work.

https://doi.org/10.1016/j.neucom.2020.02.020
0925-2312/

to the weight parameters of one base NN, and the solutions share the common objective of maximizing the classification accuracy. The advantages of this method are: (1) The MMOP solver explicitly finds a diverse set of promising solutions, which guarantees the diversity of base NNs and hence promises the generalization ability of NNE; (2) The MMOP solver can automatically identify the solution number and hence enables the self-adaptation of number of NNs used in the NNE, which ensures that the NNs to be assembled are suitable for specific case; and (3) it supports the concurrent optimization of multiple NNs in a single run, which is more time-efficient than the previous methods that repeat EAs for each individual NN.

According to the above, how to design a well-performed EA to deal with the MMOP becomes an important issue to be addressed in this study. Commonly, the niching techniques inspired from biology are incorporated into EAs to obtain niching EAs for tackling MMOPs. In biology, the organization consisting of individuals possessing the same characteristics is called species, and the environment where species reside is called niche. The niching techniques are designed to locate and preserve potential niches, maintain multiple promising solutions, and avoid the convergence to a single optimum. A number of niching methods have been introduced to solve MMOPs. Representatively, crowding [24], fitness sharing [25], speciation [26], clearing [27], clustering [28], and restricted tournament selection [29]. However, the niche identification is still a challenging task, as the size and distribution of niches depend on the specific optimization situation.

In order to develop an MMOP optimizer to find NNs in terms of accuracy and diversity, it is essential to identify niches self-adaptively and to set the EA parameters appropriately. To locate the niches, we develop a heuristic clustering method that forms the candidate niches depending on the population distribution in the specific problem landscape. The baseline EA adopted in the paper is differential evolution (DE) algorithm, owing to its advantages of having a few control parameter, high computational efficiency, and easy implementation [24,26,30–36]. DE, like other EAs, is very sensitive to the control parameters, i.e., population size *NP*, the scaling factor *F*, and the crossover rate *CR*. In the literature, most researchers focus on the adaptation of the flat parameters such as *F* and *CR* [30,35,37–39], while the research on adaptive *NP* has rarely been seen. The adaptation of *NP* is more challenging because it needs to vary the population structure. We design a parameter adaptation mechanism that is able to adapt the population size during the evolution. Moreover, an auxiliary movement scheme is developed to enhance the performance of the proposed method.

To summarize, in this paper, we model the NNE training as MMOP to realize the concurrent and efficient optimization of multiple base learners. A novel algorithm named adaptive niching DE (ANDE) is developed to accomplish the optimization task. The novelties of ANDE are summed up as follows:

1) *Heuristic Clustering Method*: We develop a clustering-based niching method to heuristically cluster the subpopulation, avoiding to set a difficult niche/cluster size. The subpopulations dominate different subregions and track multiple optimal solutions on different peaks simultaneously.
2) *Parameter Adaptation Mechanism*: The control parameters (*NP*, *F*, and *CR*) of the niching DE are adaptively adjusted at the same time. The parameter *NP* is dynamically adapted depending on the population state with a population exclusion technique and a population augmentation technique. Besides, the parameters *F* and *CR* of each individual are dynamically adjusted according to their success update distributions at each generation.
3) *Auxiliary Movement Scheme*: To further enhance the performance of the algorithm, two movement schemes are designed: a) the uniform random motion helps the stagnating subpopulation to jump out of the basin of a peak and turns the subpopulation into a converging one, and b) the Brownian movement guides the converging subpopulation to a more promising area.

Experiments demonstrate that the algorithm under discussion performs more prominently than the state-of-the-art niching algorithms on the 20 well-known multimodal optimization problems of the CEC2013 test suite. Moreover, we propose an extended ANDE for NNE training on the classification datasets. The experiments show the effectiveness and competitiveness of the ANDE-NNE.

The rest of this paper is presented as follows. Section 2 provides a brief overview of the DE algorithm and its recent development, the niching techniques for MMOPs, and researches of NNE. Section 3 introduces the implementation of ANDE. Experimental results on MMOPs are reported and analyzed in Section 4. Further, in Section 5, ANDE is applied to optimize NNE. Last, in Section 6, conclusions are reached.

2. Background and related work

2.1. Differential evolution

DE is a real-number EA suggested by Storn and Price in 1995 for solving Chebyshev polynomial problems [40]. The following description demonstrates the basic procedures of DE. First, a random initial population is created and the search range of the solution is set within $[X_{\min}, X_{\max}]$, for the jth dimension of the individual $\vec{x}_i$,

$$x_{i,j} = x_{\min j} + rand(0,1) \cdot \left(x_{\max j} - x_{\min j}\right) \tag{1}$$

where $x_{\min j}$ and $x_{\max j}$ are the lower limit and upper limit of $x_{i,j}$, and $rand(0, 1)$ is a uniformly distributed random number ranging from 0 to 1. After the initialization, three operators: mutation, crossover, and selection are employed in DE generations until the termination condition is met.

Mutation: DE generates mutation vectors by adding scalable differences between other individuals to basis individuals of the population. Three mutation strategies that are most frequently used are as follows:

1) DE/rand/1

$$\vec{v}_i = \vec{x}_{r1} + F \cdot (\vec{x}_{r2} - \vec{x}_{r3}) \tag{2}$$

2) DE/best/1

$$\vec{v}_i = \vec{x}_{best} + F \cdot (\vec{x}_{r1} - \vec{x}_{r2}) \tag{3}$$

3) DE/current-to-best/1

$$\vec{v}_i = \vec{x}_i + F \cdot (\vec{x}_{best} - \vec{x}_i) + F \cdot (\vec{x}_{r1} - \vec{x}_{r2}) \tag{4}$$

where $\vec{v}_i$ is the mutant vector of current individual; $\vec{x}_{r1}, \vec{x}_{r2}$, and $\vec{x}_{r3}$ are random samples (the indices of i, r_1, r_2, and r_3 are distinct); the control parameter F is the scaling factor in [0, 2]; and $\vec{x}_{best}$ is the best individual.

Crossover: After mutation, the trail vector $\vec{u}_i$ is formed by carrying out binomial crossover operation on the current vector $\vec{x}_i$ and mutant vector $\vec{v}_i$ with

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand(0,1) \le CR, \text{ or } j = I \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{5}$$

where CR is the crossover rate in [0, 1]; I denotes a random integer generated from [1, dim] (note that dim is the problem dimension), which ensures that at least one dimension derives $\vec{u}_i$ from $\vec{v}_i$.

Selection: The individual with better fitness value between the trail vector $\vec{u}_i$ and the current vector $\vec{x}_i$ will be selected for the next generation. For instance, if the objective optimization function is a maximization problem, the selected vector will be given by:

$$\vec{x}_i = \begin{cases} \vec{u}_i, & \text{if } f(\vec{u}_i) > f(\vec{x}_i) \\ \vec{x}_i, & \text{otherwise} \end{cases} \quad (6)$$

where $f(\vec{x})$ is the objective function to be maximized.

DE, like other EAs, suffers from the sensitive control parameter settings (i.e., *NP, F*, and *CR*). In recent years, many researchers have dedicated to adapt the control parameters, such as *F* and *CR* in [30,35,37–39]. Civicioglu et al. [41] proposed a topology-free and parameter-free DE. Besides, various techniques are incorporated into DE for a powerful capability. Xu et al. [42] strengthened DE by introducing a cooperative ranking-based mutation to solve a constrained optimization problem. In [43], Peng et al. enhanced DE with a long short-term memory to predict electricity price. Wu et al. [44] assembled various DE variants to enable cooperative search. Due to the promising search ability, DE is also used to handle the practical problems, such as the electricity energy consumption forecasting [45], the congestion management in the presence of wind turbine generators [46], and the reactive power management [47].

2.2. Niching methods for multimodal optimization

Multimodal optimization is designed to locate multiple optimal solutions (global and/or local) to the maximum in a single run. For solving MMOPs, numerous niching methods have been blended in EAs [32,48,49].

To the best of our knowledge, since Thomsen [24] firstly integrated crowding technique into DE algorithm to tackle MMOPs, a large number of niching based DE algorithms have emerged and developed rapidly. In crowding DE (CDE), each offspring individual is compared with its closest parent individual computed by Euclidean distance from a crowd consisting of randomly selected *CF* individuals. The parent individual is to be substituted if the offspring is superior in the next generation.

Later, a species-based DE algorithm (SDE) [26] was proposed, where the population is zoned as multiple subpopulations (species). Each subpopulation is constituted by individuals within the species radius centered on a species seed. However, neither of these two techniques is ideal in handling with complex optimization problems, and the performance greatly depends on the niching parameters.

Recently, a neighborhood-based method is proposed and is prevailing in the niching method design. Qu et al. [33] and Gao et al. [35] incorporated the neighborhood strategy into the population partition, which are known as the neighborhood-based DE (i.e., NCDE, NSDE, and NShDE) [33] and self-adaptive clustering-based DE (i.e., Self-CSDE and Self-CCDE) [35], respectively. Further studies reveal that after several generations of evolution, individuals would cluster around global or local individuals. Inspired by this, Li brought up fitness Euclidean-distance ratio PSO (FERPSO) [50] and ring topology PSO (R2PSO and R3PSO) [51], which are free from introducing any niching parameter. Besides, two emerging mutation strategies (DE/nrand/1 and DE/nrand/2) [52] were introduced to produce niching effect. Based on DE/nrand/1 strategy, a dynamic archive niching DE (dADE/nrand/1) was proposed in [34]. Two mechanisms are incorporated to this algorithm: the control parameter adaptation scheme in JADE [53] and the re-initialization scheme to achieve efficiently search. The DE/nrand family is very simple to implement. Besides, to alleviate the high computational cost in calculating the distance, an index-based neighborhood (DE/inrand/1R) [54] was designed based on a predefined topology. Then, Biswas et al. [36] conceived a novel parent-centric normalized mutation strategy in niching DE (PNPCDE), which utilizes neighborhood information while performing mutation, tracking and locating optimal solutions without loss of niches. Further, an improved locally informative niching DE (LoINDE) [31] was presented for relaxing the selection pressure. Similarly, Qu et al. [55] put forward a distance-based locally informed PSO (LIPS), which enhances the search ability with the information obtained from local best particles neighborhoods. Recently, the local search strategy and the dynamic cluster sizing strategy are integrated into the estimation of distribution algorithm (EDA) for crowding and speciation methods, which are named as LMCEDA and LMSEDA [56].

2.3. Neural network ensemble

NNE was formally proposed by Hansen and Salamon in 1990 [2]. They proved that they can perfect the generalization power of neural network system by training a certain number of individual NNs and synthesizing their outputs. The implementation of NNE basically focuses on two aspects, (a) how to generate a set of individual NNs and (b) how to combine the predicted outputs of NNs.

In generating individual NNs, two most typical techniques are Boosting and Bagging (Bootstrap Aggregating). The Boosting method was first proposed by Schapire [57], and later improved by Freund in 1995 [58]. This method creates a serial sequence of NNs, in which the training sample distribution of each NN is adjusted according to the performance of the network generated before it. Specifically, the training samples with errors in the current NN will have a greater probability of appearing in the next round of training sample set. At present, the most popular algorithm of boosting family is AdaBoost algorithm put forward by Freund and Schapire in 1997 [59]. The Bagging is a parallel method based on bootstrap sampling [60,61]. The training samples of each individual NN are randomly taken out from the initial training set. Therefore, the training sets of each individual NN are independent of each other. That is, in an individual training set, there may be zero or many identical samples. Generally, the sample size of the individual NN is the same as the initial training set. The final output results of NNE are obtained by combining the results of multiple individual NNs. For regression analysis, the most commonly used combination strategy of NNE is the averaging strategy. For classification problem, the majority voting is the most frequently used strategy. Fig. 1 demonstrates the framework of NNE classification derived from the Bagging algorithm.

Actually, training the individual NNs in NNE is a parameter optimization problem: how to find the appropriate parameters (e.g., the connection weights of NNs) to minimize the error between the expectation and the actual output values is the goal of the neural network optimization. The back propagation algorithm (BP) [62] originated from the gradient descent strategy is the most well-known neural network training algorithms. The role of the gradient descent strategy is to seek the optimal solution along the direction of negative gradient. A set of parameters will not be updated if the gradient is equal to zero. That is, a local optimum is found, but it is not necessarily global optimum. The possibility of falling into local optimum limits the performance of using BP to train individual NNs. So far, EAs have been widely adopted to train individual NNs of NNE [63–65] mainly because of the following two advantages. First, EAs optimize the solution of the problem in the form of group collaboration, rather than giving the strict mathematical derivation for the problem. Second, EAs provide opportunities of jumping out of local optimum and have good robustness in locating global optimum.

In spite of the efforts put into employing BP/EAs to train NNE, there are still certain disadvantages of the universal NNE methods as follows: (1) The traditional NNE cannot guarantee the diversity

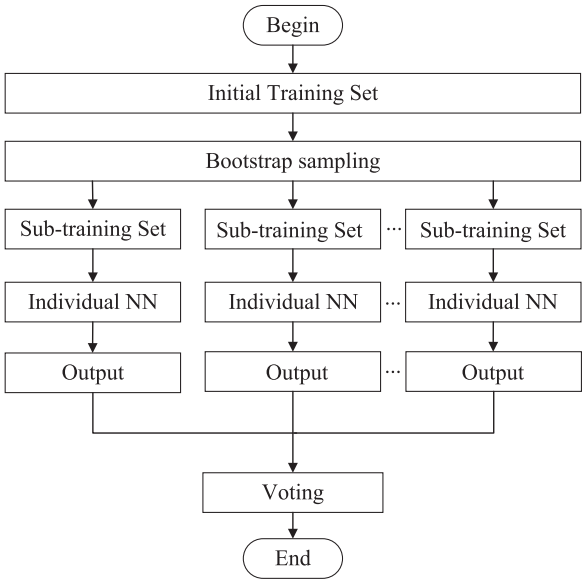


Fig. 1. The flowchart of NNE for classification problems based on bagging method.

of individual NNs, since different NNs may converge to the same optimal solution in independent evolutionary process. (2) It is a predicament to set an appropriate number of individual NNs. A relative minority of NNs may lead to the lack of diversity, while a relatively large number of NNs may cause redundancy. (3) Whether using the traditional BP or EAs method, each training only produces one NN. In order to obtain enough NNs in NNE, multiple training of a single NN needs to be performed repeatedly, which makes NNE compute at a price.

Along with the emergence of niching-based EAs, to tackle MMOPs, it is now feasible to explore and maintain different global optimal solutions simultaneously. At present, few attempts have been made to the development of niching-based EAs for training NNE, although they contribute to generate all diverse individual NNs that possess good diversity and accuracy in a single run for NNE. In [65], a niching-based PSO method (NichePSO) [66] was proposed to evolve NNE, each subpopulation is utilized to optimize the parameters that are included in each individual NN. However, this approach has limitation that the number of individual NNs needs to be appropriately adjusted based on different problems. Gong et al. proposed a Bare-Bones niching DE (BNDE) [30], which self-adapts the number of NNs, thus possessing good diversity and accuracy. It is worth noting that the parameter *NP* of BNDE simply hinges on problem dimension. Therefore, making efforts to breed a self-adaptive strategy so as to adjust *NP* over the course of evolution process is a question worth pondering.

3. The proposed niching method

3.1. Workflow of the ANDE

The proposed ANDE is realized in Algorithm 1. A population $\mathbf{P_0}$ is randomly initialized and partitioned into a set of subpopulations through the heuristic clustering method introduced in Algorithm 2. At each generation, each subpopulation performs the mutation, crossover, and selection independently, while using adaptive *F* and *CR* values. Here we choose DE/best/1 as the mutation strategy. The obtained subpopulations are classified into three categories: stagnating, converging, and converged, and subsequently perform the respective operation. The best individual in a stagnating subpopulation undergoes uniform random movement, and the best individual in a converging subpopulation undergoes Brownian movement,

Algorithm 1 ANDE

1: $t = 0$; // t is the current evolutionary generation.
2: Randomly initialize a population $\boldsymbol{P_0}$;
3: $\boldsymbol{P} \leftarrow$ Cluster($\boldsymbol{P_0}$); //cluster $\boldsymbol{P_0}$ according to **Algorithm 2**
4: **while** stopping criterion is not satisfied **do**
5: **for** each subpopulation P_k **do**
6: **for** each individual $\vec{x}_i$ **do**
7: Update F and CR according to Eqs. (11) and (12);
8: Generate a mutate vector $\vec{v}_i$ by:
$\vec{v}_i = \vec{x}_{best} + F \cdot (\vec{x}_{r1} - \vec{x}_{r2})$;
9: Generate a trial vector $\vec{u}_i$ according to Eq. (5);
10: Perform selection according to Eq. (6);
11: **end for**
12: **if** P_k is identified as a stagnating subpopulation **then**
13: Apply uniform random movement using Eq. (13);
14: **else if** P_k is identified as converging **then**
15: Apply Brownian movement using Eq. (14);
16: **else if** P_k is identified as a converged subpopulation **then**
17: Insert $\vec{x}_{best}^{P_k}$ to the archive;
18: Reinitialize the individuals P_k in and evaluate them;
19: **end if**
20: **end for**
21: Exclusion($\boldsymbol{P}$); // according to **Algorithm 3**
22: Augmentation($\boldsymbol{P}$);// according to **Algorithm 4**
23: $t \leftarrow t + 1$;
24: **end while**

Algorithm 2 Heuristic Clustering

1: **for** $k = 1 \rightarrow |\boldsymbol{P_0}|$ **do**
2: $P_k \leftarrow \boldsymbol{P_0}[k]$;
3: **end for**
4: Calculate d_{inter} and d_{intra} of $\boldsymbol{P}$; $//\boldsymbol{P} = \{P_1, P_2, \cdots P_n\}$
5: **while** $d_{\text{inter}} > d_{\text{intra}}$ **do**
6: Merge P_k and P_s where $D(P_k, P_s) = \min_{k \neq s} D(P_k, P_s)$;
7: Update d_{inter} and d_{intra};
8: **end while**

Algorithm 3 Population Exclusion

1: Select P_k and P_s,where $D(P_k, P_s) = \min_{k \neq s} D(P_k, P_s)$;
2: $f_{\min} = \min(f(\vec{p}_k), f(\vec{p}_s))$;
3: **if** $d(\vec{p}_k, \vec{p}_s) \leq d_{\text{overlapping}}$ and $f(\vec{m}) \geq f_{\min}$ **then**
4: **if** $f(\vec{x}_{best}^{P_k}) > f(\vec{x}_{best}^{P_s})$ **then**
5: remove P_s;
6: **else**
7: remove P_k;
8: **end if**
9: **end if**

for the purpose of striking a potential equilibrium between exploration and exploitation. For a converged subpopulation, its best individual will be inserted in the external archive, and all individuals in this subpopulation will be reinitialized. Afterwards, the entire population is adjusted, by eliminating inefficient individuals (the population exclusion in Algorithm 3) and adding new individuals (population augmentation in Algorithm 4). The above process is repeated until the stopping criterion is satisfied.

It can be observed from the above procedure that, in addition to the baseline DE algorithm, the basic components introduced by ANDE are: heuristic clustering, population adaptation, parameter adaptation of *F* and *CR*, auxiliary movement, and archiving. In the next of this section, we describe these basic components one by one. Note that, throughout the following descriptions, a population (or subpopulation) is recognized as converging if the number of best-so-far solutions increases or the fitness of the best-so-far solution is updated. A population (or subpopulation) is recognized as converged if its radius is under a threshold value. A population (or subpopulation) is recognized as stagnating if it shows no improvement for successive *St* generations.

Algorithm 4 Population Augmentation

```
1: G_t = 0
2: if f(x⃗^P_best) == f(x⃗^P_best)_{t-1} then
3:     for j = 1 → |P| do
4:         if f(x⃗^{P_j}_best) > f(x⃗^P_best) − θ then
5:             G_t + +;
6:         end if
7:     end for
8: end if
9: if G_t == G_{t-1} then
10:    δ + +;
11: else
12:    δ = 0;
13: end if
14: if δ == St then
15:    create a random subpopulation P′;   //|P′| = (1/M) Σ_{k=1}^{M} |P_k|
16:    P = P ∪ P′;
17:    δ = 0;
18: end if
```

3.2. Heuristic clustering method

In existing cluster-based niching algorithms, when partitioning the whole population into a set of subpopulations, the size of each subpopulation is mostly equal to a given cluster size. Nevertheless, the performance of these algorithms is really susceptible to the effect of the cluster size, which is hard to appropriately set as the cluster size is problem-dependent. To solve this problem, a heuristic clustering method is developed that it produces non-overlapping subpopulations without introducing any niching parameters.

In details, before the clustering, each individual in the population is regarded as a subpopulation. Next, the subpopulations are iteratively clustered according to the distance between subpopulations (denoted as d_{inter}) and the distance within subpopulations (denoted as d_{intra}). When the condition $d_{\text{inter}} > d_{\text{intra}}$ is satisfied, the closest pair of subpopulations will be merged into one subpopulation. In this way, a group of subpopulations without overlapping regions between each other is obtained without a difficult cluster size. Algorithm 2 presents the workflow of the heuristic clustering method to form the clustered subpopulations.

Note that the d_{inter} is the sum of inter-subpopulation distances between each pair of subpopulations, while d_{intra} is the sum of intra-subpopulation distances between each pair of individuals within the same subpopulation.

$$d_{\text{inter}} = \sum\nolimits_{P_k, P_s \in P} D(P_k, P_s) \tag{7}$$

$$d_{\text{intra}} = \sum_{\vec{x}_i, \vec{x}_r \in P_k} d(\vec{x}_i, \vec{x}_r) \tag{8}$$

where $d(\vec{x}_i, \vec{x}_r)$ is the Euclidean distance between individuals $\vec{x}_i$ and $\vec{x}_r$, k is the figure of the subpopulations of the current population $\boldsymbol{P}$. $D(P_k, P_s)$ is the distance between two subpopulations.

$$D(P_k, P_s) = \min_{\vec{x}_i \in P_k, \vec{x}_r \in P_s} d(\vec{x}_i, \vec{x}_r) \tag{9}$$

In addition, as DE must contain at least four individuals, the subpopulation with less than four individuals will be eliminated.

3.3. Population adaptation strategy

As a main control parameter of DE, the population size *NP* has a crucial influence on the performance of the algorithm. Besides, for ensemble learning, the parameter *NP* affects the number of NNs provided by the algorithm to some extent. Accordingly, the insufficient or excessive number of the provided NNs may cause poor ensemble performance or redundancy of NNE. Inspired by this, we suggest a dynamic adaptive *NP* strategy, which can effectively deal with different MMOPs through a fixed initial *NP*.

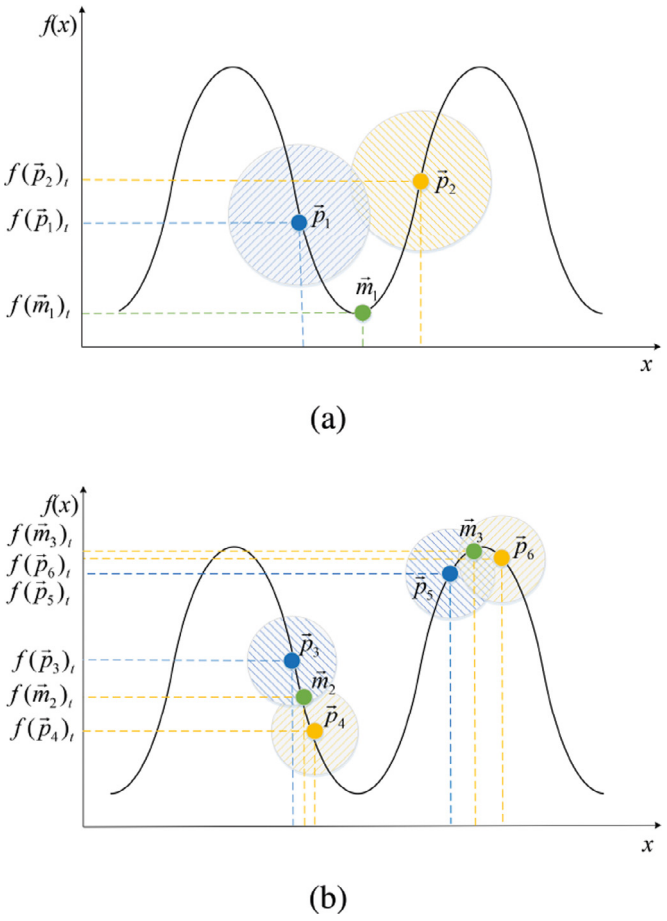


Fig. 2. Illustration of the overlapping identification method. (a) Non-overlapping subpopulations: the fitness of the midpoint $\vec{m}$ is worse than the centers of the subpopulations P_1 and P_2. (b) Overlapping subpopulations: the fitness of the midpoint $\vec{m}$ is better than one or both of the centers of the subpopulations P_1 and P_2.

To be specific, the value of *NP* is adaptively adjusted (increase, decrease, and maintain) at different phases of evolution. In this way, the algorithm can efficiently locate and track multiple optima according to the current problem landscape.

(1) *Population exclusion*

In our proposed algorithm, if two subpopulations are overlapped on the same peak, the redundant subpopulation will be removed so as to save fitness evaluations (FEs) and to perform further exploration. In order to detect the overlapping, we utilize a topology-based method namely Hill-Valley that is proposed by Ursem [67,68] to examine the landscape topography. As shown in Fig. 2(a), P_1 and P_2 are two subpopulations, and $\vec{p}_1$ and $\vec{p}_2$ are their centers, which are marked with blue and yellow dots, respectively. The midpoint of $\vec{p}_1$ and $\vec{p}_2$, denoted as $\vec{m}$, is marked with a green dot. Mathematically, the center $\vec{p}_k$ of the subpopulation P_k is calculated by

$$\vec{p}_k = \sum_{\vec{x}_i \in P_k} \frac{\vec{x}_i}{|P_k|}. \tag{10}$$

In the case of Fig. 2(a), the fitness value of $\vec{m}_1$ is worse than that of both $\vec{p}_1$ and $\vec{p}_2$. Therefore, the two subpopulations, P_1 and P_2 are not overlapping as they explore on different peaks. Fig. 2(b) shows the overlapping subpopulations in two occasions: the fitness value of the midpoint $\vec{m}_2$ is between that of $\vec{p}_3$ and $\vec{p}_4$; the fitness value of the midpoint $\vec{m}_3$ is better than that of both $\vec{p}_5$ and $\vec{p}_6$.

In this way, we can identify the overlapping subpopulations that search on the same peaks.

The overlapping subpopulations waste unnecessary search efforts on the same peak, and therefore we perform the population exclusion strategy to avoid the redundant search, which is presented as follows. At each generation, a pair of subpopulations with the nearest distance is selected. Note that the distance between the two subpopulations is smaller than the threshold distance $d_{\text{overlapping}} = 0.01$, the overlapping identification method will be conducted. If they are identified as overlapping, the subpopulation with worse best fitness value will be removed from $\boldsymbol{P}$. Algorithms 3 shows the pseudocode of population exclusion, where the $f(\vec{x}^{P_k}_{best})$ is the fitness value of the best individual of P_k.

(2) *Population augmentation*

The population augmentation method adds a new subpopulation to diversify the stagnating population. First, we test whether the fitness value of the best-so-far solution of the entire population at the current generation (denoted as $f(\vec{x}^{\boldsymbol{P}}_{best})$) is the same as that at the last generation (denoted as $f(\vec{x}^{\boldsymbol{P}}_{best})_{t-1}$). If the condition is true, we use a variable G_t to count the total number of the optima found by all the subpopulations. The best solution of a subpopulation is identified as an "optimum" when its values is equal to the $f(\vec{x}^{\boldsymbol{P}}_{best})$ or the difference between the solution and $f(\vec{x}^{\boldsymbol{P}}_{best})$ is smaller than a threshold value θ. If the value of G_t is the same as G_{t-1}, we increase the counter δ by one. The current population $\boldsymbol{P}$ is regarded as a stagnating population when the value of G_t remains unchanged over St successive generations. To increase the diversity of the population, an extra subpopulation P' is created and added to the population $\boldsymbol{P}$. The size of P' is equal to the average subpopulation size of all subpopulations in current generation. We choose a relatively small number 0.001 for θ and set St to 3 in this strategy. The population augmentation is shown in Algorithm 4.

3.4. Parameter adaption of F and CR

The other two control parameters of DE, F and CR, are fairly sensitive to the performance of the algorithm. Traditional DE algorithms set fixed values of F and CR. For the purposes of making the selection of these parameters independent of the optimization problem, we adopt the method in [53] to self-adjust the parameters. We associate each individual with an F_i and a CR_i. The parameters are updated according to different distributions: $F_i \sim Cauthy(\mu_F, 0.1)$ and $CR_i \sim N(\mu_{CR}, 0.1)$. The corresponding mean values μ_F and μ_{CR} are both initialized as 0.5 and updated at each generation by

$$\mu_F = (1-b)\cdot \mu_F + b \cdot mean_L(S_F) \tag{11}$$

$$\mu_{CR} = (1-b)\cdot \mu_{CR} + b \cdot mean_A(S_{CR}) \tag{12}$$

where b is a constant that is usually set to 0.1; S_F and S_{CR} are the sets of all F_i and CR_i that successfully improve the trial vectors, respectively; $mean_L(\cdot)$ and $mean_A(\cdot)$ denote the Lehmer mean and arithmetic mean, respectively.

3.5. Auxiliary movement scheme

For a stagnating subpopulation P_k, whose best individual stops improving for St successive generations, we try to turn the P_k into a converging subpopulation by performing a uniform random motion on the best individual of P_j

$$\overrightarrow{x'}^{P_k}_{best} = U[\vec{x}_{\min}, \vec{x}_{\max}] \tag{13}$$

where $U[a, b]$ is an equiprobability distribution within the interval $[a, b]$.

Simultaneously, for a converging subpopulation P_k, to reduce the possibility of premature convergence of the algorithm and to increase subpopulation diversity, the best individual in P_k is selected to perform a Brownian movement

$$\overrightarrow{x'}^{P_k}_{best} = N(\vec{x}^{P_k}_{best}, R(P_k)) \tag{14}$$

where $\vec{x}^{P_k}_{best}$ is the best individual of P_k, and $R(P_k)$ is the radius of P_k, i.e., the average distance of the centroid individual p_k and other individuals in the subpopulation.

After the movements, the $\overrightarrow{x'}^{P_k}_{best}$ will replace $\vec{x}^{P_k}_{best}$ if $f(\overrightarrow{x'}^{P_k}_{best})$ is better than $f(\vec{x}^{P_k}_{best})$. The uniform random motion makes it possible for the subpopulation which is falling into the basin of an explored peak to jump to a more promising area. The Brownian movement may lead the best individual of the subpopulation to a better position. It is worth noting that, after performing movements, if the generated individuals are outside the predefined variable boundaries, they are reset to the feasible boundary.

3.6. Archiving technique

During the evolution, the population to be reinitialized or excluded will be archived. Particularly, for a converged subpopulation, the individuals have totally gathered at the summit of a peak. It is unnecessary to continue exploiting on the peak that would result in waste of FEs. To avoid the unnecessary search on such subpopulation, the best individual of the converged subpopulation will be stored in an external archive. The converged subpopulation will be reinitialized to increase the population diversity and search for unexplored regions of the problem space. We adopt the archiving techniques in [34] to determine the qualification of the candidate solutions and selectively collect those promising solutions.

3.7. Complexity analysis

As niching EAs involve the fitness evaluation and distance measure, we consider these two operations as unit operations with $O(1)$ complexity for simplicity. In the following, we discuss the computational complexity of each major component of ANDE in each generation. Note that, as the subpopulation size and the subpopulation number vary along the evolution, we take the expected value to conduct the calculation. The expected subpopulation size and the expected subpopulation number are denoted as NP and M, and the problem dimension is denoted as dim. The heuristic clustering in Algorithm 2 consumes $O(NP^2)$ to calculate the distance measures, i.e., d_{inter} and d_{intra}. For the population exclusion method in Algorithm 3, the most costly operation is to select two nearest subpopulation with $O(NP^2)$ complexity. Algorithm 4 takes $O(M \times dim)$ complexity to reinitialize the stagnating subpopulation if necessary. Then, the adaptive mutation and crossover operations cost $O(NP \times dim)$. The auxiliary movement scheme takes $O(M \times dim)$ to conduct the movement on the best solutions of the stagnating and converging subpopulations. The archiving technique consumes $O(AN)$ to add a solution, where AN is the solution number of the archive. Ideally, the most costly complexity is $O(NP^2)$. Therefore, the overall computational complexity of ANDE is $O(NP^2)$, which is consistent with the complexity of most niching EAs.

4. Numerical studies for MMOPs

The key to developing an NNE with good generalization performance is to choose or design a competitive niching-based EA for the training of individual NNs in NNE. That is, the niching-based

Table 1
Charcteristics of the 20 benchmark functions and parameter settings.

Function	Dimension	No.global optima	*NP*	MaxFEs
*F*1	1	2	100	5.00E+04
*F*2	1	5	100	5.00E+04
*F*3	1	1	100	5.00E+04
*F*4	2	4	100	5.00E+04
*F*5	2	2	100	5.00E+04
*F*6	2	18	100	2.00E+05
*F*7	2	36	300	2.00E+05
*F*8	3	81	300	4.00E+05
*F*9	3	216	300	4.00E+05
*F*10	2	12	100	2.00E+05
*F*11	2	6	200	2.00E+05
*F*12	2	8	200	2.00E+05
*F*13	2	6	200	2.00E+05
*F*14	3	6	200	4.00E+05
*F*15	3	8	200	4.00E+05
*F*16	5	6	200	4.00E+05
*F*17	5	8	200	4.00E+05
*F*18	10	6	200	4.00E+05
*F*19	10	8	200	4.00E+05
*F*20	20	8	200	4.00E+05

EA used in NNE should be good at (1) seeking for more global optimal solutions, (2) upgrading the accuracy of global optimal solutions, and (3) speeding up the convergence. Therefore, the experiments in this section are conducted to verify the above advantages of the proposed algorithm.

4.1. Experimental setup and evaluation protocols

In this section, we verify the performance of the proposed algorithm through 20 multimodal test functions of the CEC2013 benchmark test suite [69]. The employed functions are of three categories. The first ten functions belong to well-known low-dimensional fundamental functions. The next five functions are low-dimensional composition functions with many local optimal solutions. The last five functions are high-dimensional composition functions. The dimension and the number of global optimal solutions of these functions are described in Table 1. For more detailed information about this test suite, please refer to [69].

Then the experimental results of ANDE are compared with those of the following 13 commonly compared or the latest multimodal optimization algorithms:

1) CDE [24]: crowding-based DE;
2) SDE [26]: species-based DE;
3) NCDE [33]: neighborhood-based CDE;
4) NSDE [33]: neighborhood-based SDE;
5) dADE/nrand/1 [34]: a dynamic archive niching DE with parameter adaptation technique;
6) DE/inrand/1R [54]: DE with index-based ring neighborhood;
7) PNPCDE [36]: a proximity-based CDE with parent-centric normalized mutation;
8) LoISDE [31]: a local information sharing DE;
9) FERPSO [50]: a fitness euclidean-distance PSO;
10) R3PSO [51]: an lbest PSO with a ring-3 topology neighborhood;
11) LIPS [55]: a distance-based locally informed PSO;
12) LMCEDA [56]: local search-based multimodal EDA, adopting selection operation in CDE;
13) LMSEDA [56]: local search-based multimodal EDA, adopting selection operation with niches;

All the algorithms are terminated when the maximum number of FEs (MaxFEs) are exhausted. The specific MaxFEs and the popu-

Table 2
PR and SR of ANDE on 20 benchmark functions.

	*F*1		*F*2		*F*3		*F*4		*F*5	
Accuracy Level ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1.0E-01	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
1.0E-02	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
1.0E-03	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
1.0E-04	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
1.0E-05	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
	*F*6		*F*7		*F*8		*F*9		*F*10	
Accuracy level ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1.0E-01	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
1.0E-02	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	0.790	0.000	**1.000**	1.000
1.0E-03	**1.000**	1.000	0.994	0.800	**1.000**	1.000	0.712	0.000	**1.000**	1.000
1.0E-04	**1.000**	1.000	0.950	0.200	**1.000**	1.000	0.642	0.000	**1.000**	1.000
1.0E-05	**1.000**	1.000	0.928	0.050	**1.000**	1.000	0.613	0.000	**1.000**	1.000
	*F*11		*F*12		*F*13		*F*14		*F*15	
Accuracy level ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1.0E-01	**1.000**	1.000	0.775	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
1.0E-02	0.787	0.000	0.750	0.000	0.667	0.000	0.667	0.000	0.650	0.000
1.0E-03	0.667	0.000	0.750	0.000	0.667	0.000	0.667	0.000	0.650	0.000
1.0E-04	0.667	0.000	0.750	0.000	0.667	0.000	0.667	0.000	0.635	0.000
1.0E-05	0.667	0.000	0.750	0.000	0.667	0.000	0.667	0.000	0.625	0.000
	*F*16		*F*17		*F*18		*F*19		*F*20	
Accuracy level ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1.0E-01	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	0.475	0.400
1.0E-02	0.667	0.000	0.425	0.000	0.700	0.000	0.825	0.000	0.305	0.000
1.0E-03	0.667	0.000	0.425	0.000	0.667	0.000	0.400	0.000	0.275	0.000
1.0E-04	0.667	0.000	0.425	0.000	0.667	0.000	0.350	0.000	0.250	0.000
1.0E-05	0.667	0.000	0.350	0.000	0.633	0.000	0.273	0.000	0.050	0.000

Table 3
PR and SR on 20 benchmark functions at accuracy level ε = 1.0E-04.

	ANDE		CDE		SDE		NCDE		NSDE		dADE/nrand/1		DE/inrand/1R	
Func	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
*F*1	**1.000**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*2	**1.000**	1.000	**1.000(** ≈ **)**	1.000	0.920(+)	0.800	**1.000(** ≈ **)**	1.000	0.856(+)	0.544	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*3	**1.000**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*4	**1.000**	1.000	**1.000(** ≈ **)**	1.000	0.366(+)	0.000	**1.000(** ≈ **)**	1.000	0.822(+)	0.436	**1.000(** ≈ **)**	1.000	**0.980(** ≈ **)**	0.920
*F*5	**1.000**	1.000	**1.000(** ≈ **)**	1.000	0.910(+)	0.824	**1.000(** ≈ **)**	1.000	0.865(+)	0.632	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*6	**1.000**	1.000	0.988(≈)	0.920	0.129(+)	0.000	0.455(+)	0.000	0.176(+)	0.000	0.976(≈)	0.760	0.900(+)	0.260
*F*7	**0.950**	0.200	0.754(+)	0.000	0.122(+)	0.000	0.882(+)	0.000	0.238(+)	0.000	0.827(+)	0.000	0.320(+)	0.000
*F*8	**1.000**	1.000	0.010(+)	0.000	0.176(+)	0.000	0.020(+)	0.000	0.204(+)	0.000	0.962(+)	0.360	0.294(+)	0.000
*F*9	**0.642**	0.000	0.467(+)	0.000	0.231(+)	0.000	0.539(+)	0.000	0.166(+)	0.000	0.422(+)	0.000	0.088(+)	0.000
*F*10	**1.000**	1.000	**1.000(** ≈ **)**	1.000	0.158(+)	0.000	0.856(+)	0.720	0.357(+)	0.000	**1.000(** ≈ **)**	1.000	0.995(≈)	0.000
*F*11	0.667	0.000	0.322(+)	0.000	0.334(+)	0.000	0.788(−)	0.045	0.482(+)	0.000	0.667(≈)	0.000	0.667(≈)	0.940
*F*12	0.750	0.000	0.060(+)	0.000	0.306(+)	0.000	0.179(+)	0.000	0.364(+)	0.000	0.700(+)	0.000	0.678(+)	0.000
*F*13	0.667	0.000	0.134(+)	0.000	0.312(+)	0.000	0.667(≈)	0.000	0.276(+)	0.000	0.667(≈)	0.000	0.663(≈)	0.020
*F*14	**0.667**	0.000	0.018(+)	0.000	0.233(+)	0.000	**0.667(** ≈ **)**	0.000	0.258(+)	0.000	**0.667(** ≈ **)**	0.000	0.637(+)	0.000
*F*15	0.635	0.000	0.007(+)	0.000	0.025(+)	0.000	0.325(+)	0.000	0.189(+)	0.000	0.619(+)	0.000	0.453(+)	0.000
*F*16	**0.667**	0.000	0.000(+)	0.000	0.124(+)	0.000	**0.667(** ≈ **)**	0.000	0.184(+)	0.000	**0.667(** ≈ **)**	0.000	0.587(+)	0.000
*F*17	**0.425**	0.000	0.000(+)	0.000	0.128(+)	0.000	0.187(+)	0.000	0.176(+)	0.000	0.370(+)	0.000	0.275(+)	0.000
*F*18	**0.667**	0.000	0.189(+)	0.000	0.057(+)	0.000	0.503(+)	0.000	0.377(+)	0.000	0.623(+)	0.000	0.423(+)	0.000
*F*19	0.350	0.000	0.000(+)	0.000	0.209(+)	0.000	0.346(≈)	0.000	0.089(+)	0.000	0.029(+)	0.000	0.135(+)	0.000
*F*20	**0.250**	0.000	0.000(+)	0.000	0.000(+)	0.000	0.220(+)	0.000	0.139(+)	0.000	0.002(+)	0.000	0.152(+)	0.000
+ (ANDE is significantly better)			13		18		10		18		9		12	
− (ANDE is significantly worse)			0		0		1		0		0		0	
≈ (There is no significant difference)			7		2		9		2		11		8	

	PNPCDE		LoISDE		FERPSO		R3PSO		LIPS		LMCEDA		LMSEDA	
Func	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
*F*1	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*2	**1.000(** ≈ **)**	1.000	0.467(+)	0.250	0.793(+)	0.231	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*3	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*4	**1.000(** ≈ **)**	1.000	0.560(+)	0.000	0.241(+)	0.020	0.952(+)	0.870	0.984(≈)	0.960	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*5	**1.000(** ≈ **)**	1.000	0.705(+)	0.137	0.673(+)	0.380	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000	**1.000(** ≈ **)**	1.000
*F*6	0.659(+)	0.000	0.087(+)	0.000	0.503(+)	0.000	0.716(+)	0.000	0.672(+)	0.000	0.899(+)	0.830	0.966(+)	0.720
*F*7	0.874(+)	0.000	0.067(+)	0.000	0.243(+)	0.000	0.452(+)	0.000	0.375(+)	0.000	0.727(+)	0.000	0.611(+)	0.000
*F*8	0.260(+)	0.000	0.135(+)	0.000	0.265(+)	0.000	0.417(+)	0.000	0.064(+)	0.000	0.379(+)	0.000	0.626(+)	0.000
*F*9	0.463(+)	0.000	0.082(+)	0.000	0.074(+)	0.000	0.210(+)	0.000	0.206(+)	0.000	0.267(+)	0.000	0.227(+)	0.000
*F*10	**1.000(** ≈ **)**	1.000	0.097(+)	0.000	0.078(+)	0.000	0.870(+)	0.162	0.820(+)	0.000	**1.000(** ≈ **)**	1.000	0.990(+)	0.940
*F*11	0.660(≈)	0.000	0.231(+)	0.000	0.306(+)	0.000	0.722(−)	0.000	**0.956(−)**	0.832	0.667(≈)	0.000	0.902(−)	0.448
*F*12	0.000(+)	0.000	0.265(+)	0.000	0.275(+)	0.000	0.520(+)	0.000	0.603(+)	0.000	0.750(≈)	0.000	**0.890(−)**	0.980
*F*13	0.503(+)	0.000	0.173(+)	0.000	0.147(+)	0.000	0.641(+)	0.000	**0.712(−)**	0.167	0.667(≈)	0.000	0.667(≈)	0.000
*F*14	0.617(+)	0.000	0.165(+)	0.000	0.179(+)	0.000	0.657(+)	0.000	0.621(+)	0.000	0.667(≈)	0.000	0.667(≈)	0.000
*F*15	0.237(+)	0.000	0.167(+)	0.000	0.216(+)	0.000	0.224(+)	0.000	0.322(+)	0.000	**0.692(−)**	0.000	0.657(≈)	0.000
*F*16	0.000(+)	0.000	0.143(+)	0.000	0.050(+)	0.000	0.623(+)	0.000	0.280(+)	0.000	**0.667(** ≈ **)**	0.000	**0.667(** ≈ **)**	0.000
*F*17	0.000(+)	0.000	0.167(+)	0.000	0.042(+)	0.000	0.088(+)	0.000	0.142(+)	0.000	0.412(≈)	0.000	0.463(≈)	0.000
*F*18	0.229(+)	0.000	0.072(+)	0.000	0.063(+)	0.000	0.096(+)	0.000	0.108(+)	0.000	0.648(+)	0.000	0.650(≈)	0.000
*F*19	0.000(+)	0.000	0.047(+)	0.000	0.055(+)	0.000	0.043(+)	0.000	0.000(+)	0.000	0.421(−)	0.000	**0.452(−)**	0.000
*F*20	0.000(+)	0.000	0.066(+)	0.000	0.000(+)	0.000	0.075(+)	0.000	0.000(+)	0.000	0.082(+)	0.000	0.246(≈)	0.000
+	13		18		18		15		13		6		5	
−	0		0		0		1		2		2		3	
≈	7		2		2		4		5		12		12	

lation size *NP* for each test function are presented in Table 1. Other parameters are set according to their publications. The proposed ANDE employs the initial population $NP = 100$. Furthermore, owing to the stochastic characteristic of the algorithms, we run the algorithms independently for 50 times.

Three popular evaluation protocols in [69], i.e., peak ratio (PR), success rate (SR), and convergence speed (CS), are selected to make comparisons with ANDE and other niching algorithms. Given a fixed MaxFEs and a specific accuracy level ε, PR denotes the average percentage of the number of global optimal solutions found out of the number of known global optimal solutions. SR is defined as the percentage of successful runs out of overall runs. Here, a successful run refers to the situation that all known global optimal solutions are found. Besides, CS is the average number of FEs needed to be a successful run. Note that supposing the algorithm is unable to find out all the global optimal solutions until the MaxFEs exhausted, then the MaxFEs is used to calculate CS.

In the experiments, the five accuracy levels, i.e., $\varepsilon = 1.0E-01$, $\varepsilon = 1.0E-02$, $\varepsilon = 1.0E-03$, $\varepsilon = 1.0E-04$, and $\varepsilon = 1.0E-05$, are used to estimate the power of a niching algorithm to find multiple global optimal solutions. Due to the space limitation, in this paper, unless otherwise stated, we mainly give notice to the results at $\varepsilon = 1.0E-04$, which is common in [33,35,51,70,71].

Table 4
CS on functions *F*1-*F*5 at all five accuracy levels

Func	ANDE	CDE	SDE	NCDE	NSDE	dADE/nrand/1	DE/inrand/1R
*F*1	**1.59E+02**	1.61E+02(−)	3.66E+02(+)	5.98E+02(+)	5.72E+02(+)	1.76E+02(+)	1.85E+02(+)
*F*2	**6.32E+02**	3.56E+03(+)	1.47E+04(+)	1.36E+03(+)	2.21E+05(+)	1.87E+03(+)	1.94E+03(+)
*F*3	7.58E+02	3.71E+03(+)	1.16E+03(+)	9.73E+02(+)	1.03E+03(+)	1.17E+03(+)	7.60E+03(≈)
*F*4	8.43E+03	2.75E+04(+)	5.00E+04(+)	**5.57E+03**(−)	2.37E+04(+)	1.39E+04(+)	1.68E+04(+)
*F*5	**3.12E+03**	1.43E+04(+)	1.51E+04(+)	3.14E+03(≈)	1.93E+04(+)	4.96E+03(+)	3.45E+03(+)
+ (ANDE is significantly better)		4	5	3	5	5	4
− (ANDE is significantly worse)		0	0	1	0	0	0
≈ (There is no significant difference)		1	0	1	0	0	1
Func	PNPCDE	LoISDE	FERPSO	R3PSO	LIPS	LMCEDA	LMSEDA
*F*1	1.60E+02(−)	1.70E+02(+)	4.78E+02(+)	3.04E+02(+)	1.87E+02(+)	3.21E+02(+)	3.54E+02(+)
*F*2	2.65E+03(+)	2.30E+04(+)	3.97E+04(+)	2.76E+03(+)	1.93E+03(+)	1.43E+03(+)	1.20E+03(+)
*F*3	2.34E+03(+)	1.49E+03(+)	2.77E+03(+)	3.47E+03(+)	8.62E+02(+)	**6.92E+02**(−)	7.63E+02(≈)
*F*4	3.76E+04(+)	5.00E+04(+)	3.53E+04(+)	2.45E+04(+)	1.77E+04(+)	1.32E+04(+)	7.82E+03(−)
*F*5	8.79E+03(+)	3.16E+04(+)	4.45E+04(+)	1.02E+04(+)	4.01E+03(+)	6.21E+03(+)	3.72E+03(+)
+	4	5	5	5	5	4	3
−	0	0	0	0	0	1	1
≈	1	0	0	0	0	0	1

4.2. Experimental results and comparisons

The results of ANDE on *F*1-*F*20 in terms of PR and SR at all five accuracy levels are displayed in Table 2. The PR that equal to 1 are emphasized in **boldface**. In the table, ANDE can successfully find out all the optimal solutions at the low-dimensional test functions (i.e., *F*1 - *F*10) except for *F*9 at five accuracy levels. Specially, *F*9 is a difficult problem with 216 optima. As to the composite test functions (i.e., *F*11 - *F*20), ANDE can locate the whole optimal solution set at the accuracy level $\varepsilon = 1.0E-01$ except for *F*20. The experimental result implies the effectiveness of ANDE on the CEC2013 benchmark test suite.

To further compare the performance of ANDE with other multimodal optimization algorithms, Table 3 presents the comparison outcomes in regard to PR, SR at the accuracy level $\varepsilon = 1.0E-04$. For clarity, we mark the PRs that exhibit the best performance in **boldface**. Meanwhile, the Wilcoxon rank-sum test [72] at $\alpha = 0.05$ in relation to PR is conducted to reveal the statistical significance between ANDE and other niching algorithms. The symbols represent that ANDE gains significantly better (+), significantly worse (−) or similar (≈) results than or to the comparison algorithms. As shown in Table 3, the proposed ANDE algorithm can obtain the best PRs for 15 out of 20 test functions and successfully locate all the known optima for 8 out of 20 test functions. Moreover, ANDE performs significantly better than or achieves a similar performance with the comparison algorithms on a fair number of test functions (at least 8 out of 20). It is worth to mention that *F*9 possesses 216 optimal solutions. For the most niching EAs with a fixed population size *NP*, they can only locate a small portion of optima set. However, ANDE adopts the population adaptation mechanism with an archiving technique, which enables to find out optima whose size is more than the fixed initial *NP* (ANDE with initial $NP = 100$ can averagely find 138 optima on *F*9).

For comparison on CS, the simple functions *F*1-*F*5 are carried out as the experiment subject. As we can see in Table 4, ANDE is significantly superior to all the comparison algorithms on test functions 3 out of 5. Therefore, ANDE has a fast convergence speed for tracking and locating multiple optima.

Further, we give a more straightforward perspective on ANDE. Fig. 3 visualizes the fitness landscape of ten test functions and the final distribution of the found global optimal individuals at the accuracy level $\varepsilon = 1.0E-04$. We can see that ANDE can conduct effective search on the fitness landscape with different characteristics (uneven, rugged, complex, with many local optima).

Furthermore, the boxplot of the average execution time of ANDE and comparison algorithms on 20 test functions are drawn in Fig. 4, where the average overall running time of ANDE is similar to that of most of the comparison algorithms that they consume around 17 seconds. The outliers of the algorithms are on *F*20, which has a relative expensive fitness calculation. Although ANDE implements an adaptive parameter method, its time cost is nearly the same as other niching EAs', which is consistent with the time complexity discussion in Section 3.7.

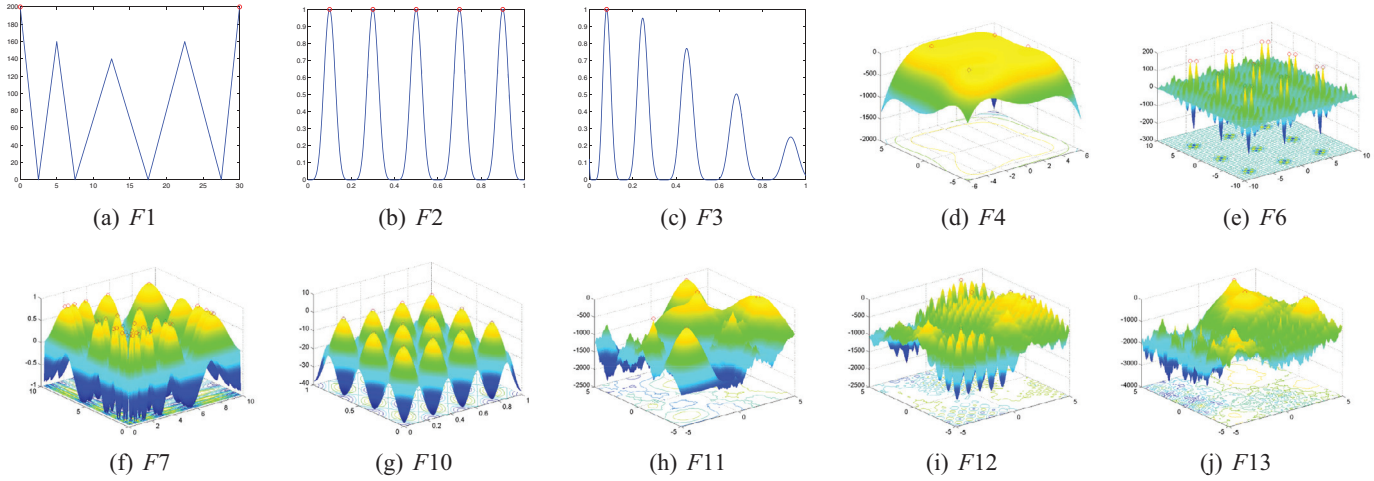
In summary, a conclusion can be reached that ANDE surpasses the most compared niching algorithms in evaluation protocols (PR, SR, and CS) with nearly the same time expense.

4.3. Effectiveness of the components of ANDE

Different components play different roles in the proposed ANDE: the parameter adaptation mechanism to adapt the control parameters to the environment and the auxiliary movement scheme to handle stagnation/converging situation for performance enhancement. We investigate the effectiveness of each component with a Wilcoxon rank-sum test at $\alpha = 0.05$. Four ANDE variants are conducted the significance test with ANDE. To be specific, "w/o augment" denotes the ANDE variant without population augmentation, "w/o exclusion" denotes the ANDE variant without population exclusion, "w/o FCRadapt" denotes the ANDE variant with fixed *F* and *CR* parameters, and "w/o move" denotes the ANDE variant without auxiliary movement scheme. The significance test results are tabulated in Table 5. From the table, we can see that ANDE significantly outperforms its variants 6, 3, 9, and 8, out of 20. The components of ANDE together contribute a powerful optimization ability.

Table 5
Significance test results of ANDE and its variants.

	w/o augment	w/o exclusion	w/o FCRadapt	w/o move
+	6	3	9	8
−	0	0	0	0
≈	14	17	11	12



(a) $F1$ (b) $F2$ (c) $F3$ (d) $F4$ (e) $F6$

(f) $F7$ (g) $F10$ (h) $F11$ (i) $F12$ (j) $F13$

Fig. 3. Visualization of the final distribution of the found global optimal solutions in the fitness landscape at the accuracy level $\varepsilon = 1.0E-04$.

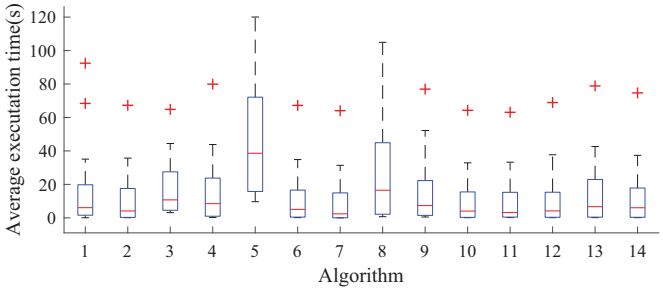
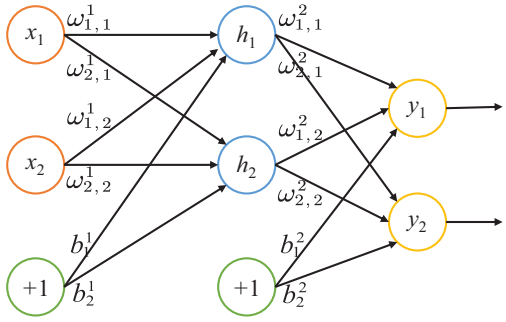


Fig. 4. The boxplot of the average execution time of the algorithms on 20 test functions of CEC2013 benchmark (figures in the horizontal axis demonstrate the different algorithms: 1- ANDE, 2- CDE, 3- SDE, 4- NCDE, 5- NSDE, 6- dADE/nrand/1, 7- DE/inrand/1R, 8- PNPCDE, 9- LoISDE, 10- FERPSO, 11- R3PSO, 12- LIPS, 13 - LMCEDA, 14 - LMSEDA). From the figure, we can observe that most of the algorithms have similar average running time.



	ω	b
$l=1$	$\omega^1_{1,1}$ $\omega^1_{2,1}$ $\omega^1_{1,2}$ $\omega^1_{2,2}$	b^1_1 b^1_2
$l=2$	$\omega^2_{1,1}$ $\omega^2_{2,1}$ $\omega^2_{1,2}$ $\omega^2_{2,2}$	b^2_1 b^2_2

Fig. 5. A neural network model with a three-layer structure. The upper subgraph illustrates the structure model and the lower subgraph tabulates the weights and biases to be optimized.

5. ANDE for training neural network ensemble

From all the experiment results above, ANDE shows a stable and efficient search performance for MMOPs. In this section, ANDE is applied for training the NNE (namely, ANDE-NNE). We first introduce ANDE-NNE. Then, we investigate the effect of ANDE-NNE by providing comparative experimental results with other neural network training methods. Finally, further analysis of ANDE-NNE is taken.

5.1. The ANDE-NNE method

The proposed ANDE ensures the diversity of NNs to enhance the generalization performance of NNE. Besides, ANDE optimizes NN with adaptive control parameters to promise the accuracy of NN. Moreover, ANDE enables parallel optimization of multiple NNs to improve the training efficiency of NNE and adaptively obtains NNs for a specific problem to avoid redundant or excessive individual NNs.

In ANDE, an individual is represented as a floating-point number vector. Accordingly, the weight parameters of an NN are treated as the variables of the individual of ANDE. Given an example of a three-layer NN illustrated in Fig. 5, the NN is composed with three layers: an input layer with two neurons and a bias, a hidden layer with two neurons and a bias, and an output layer with two neurons (note that the output neurons are one-hot encoding). The twelve connection weights of the NN are tabulated under the NN structure in Fig. 5. To utilize ANDE for NN optimization, the twelve weights are sequentially encoded into the vector of twelve dimensions. Besides, the classification accuracy is regarded as the objective function of ANDE. Generally, the number of weights (denoted as *dim*) of NN is formulated as

$$dim = \sum_{l=1}^{L-1}(nh_{l-1}+1)\times nh_l \tag{15}$$

where L denotes the number of layers and nh_l denotes the number of weights in the lth layer (note that index of layer starts from zero). In the paper, we adopt a three-layer NN with a hidden layer of five neurons, and specify the number of neurons of the input and output layers with a specific dataset. Consequently, the weight numbers varies from 32 (on Haberman dataset) to 187 (on Ionosphere dataset).

Most classification scenario can adopt ANDE-NNE to handle. For example, the Diabetes dataset (introduced in the next part) is derived from the diabetes patient records in the real world. The attributions of the patients are recorded and utilized as the input data for ANDE-NNE training. The optimized ANDE-NNE can be adopted as a classifier to predict whether a patient is at the risk of diabetes.

Table 6
Charcteristics of the 12 classification datasets.

Datasets	No. samples	No. features	No. classes
Haberman	306	3	2
Heart	270	13	2
Diabetes	768	8	2
Ionosphere	351	34	2
Wisconsin	683	9	2
Wine	178	13	3
Balance	625	4	3
Hayes-roth	132	5	3
Iris	150	4	3
Seeds	210	7	3
Car	1768	6	4
Zoo	101	17	7

5.2. Experimental setup

To weigh up the effectiveness of ANDE-NNE, twelve commonly used classification instances from the UCI machine learning repository [73] are examined. The characteristics of these datasets are described in Table 6. In the period of data preprocessing, all features are normalized to [0, 1] utilizing min-max normalization method.

We compare the results obtained by ANDE-NNE with those obtained by five neural network training methods, i.e., DE-NN, DE-NNE, BP-NN, BP-NNE, and MCCS (multi-population co-evolution chaotic searching algorithm). To clarify, DE-NN is short for using classic DE to train single NN, while DE-NNE is the ensemble of DE-NNs. BP-NN is short for using back propagation to train single NN, while BP-NNE is the ensemble of BP-NNs. MCCS [22] is an NNE optimization algorithm that utilizes three EAs (i.e., differential evolution, particle swarm optimization, and artificial bee colony algorithm) to conduct three heterogeneous subpopulation searches. Their parameters are set according to the publications. All the optimizers adopt the same NN model, i.e., three layers with one hidden layer of five neurons. The comparison algorithms give 20 individual NNs, while ANDE provides a maximum individual NNs of 20. Then, the final classification results of NNs are synthesized via the majority voting, except for MCCS that votes with different vote weights. Ten-fold cross validation method is adopted to test the performance of the algorithms. In the ten-fold cross validation, the dataset is randomly partitioned into ten equal size sub-datasets. Nine sub-datasets are reserved for model training and the rest one sub-dataset for validation. Then, the cross-validation is repeated ten times, with each sub-dataset being used exactly once as the validation data. The ten results are averaged to yield an estimation of validation accuracy. In this way, we are capable of measuring the performance of NNE methods.

5.3. Comparison experiments

Table 7 exhibits the mean values and standard deviations of the validation accuracy on datasets. Particularly, the Wilcoxon rank-sum test results with $\alpha = 0.05$ on each dataset are presented, where the symbol "+" denotes that ANDE-NNE significantly outperforms the comparison algorithm, the symbol " $\approx$ " indicates no significant difference between them, and "−" implies that ANDE is significantly worse than the rival. Besides, the best values are marked in boldface. We can see from the table that ANDE-NNE performs the best on the most datasets, 10 out of 12. ANDE-NNE obtains small standard deviations that is averaged at 5.87, while the average standard deviations of the comparison algorithms are larger than 8.4. ANDE-NNE significantly outperforms or shows no significant difference with the comparison algorithm. Besides, BP and DE perform better than their corresponding NNE methods in terms of average validation accuracy. BP-NNE and DE-NNE simply aggregate NNs without considering their diversity, which impairs the generation ability of their NNE methods. MCCS evolves three heterogeneous populations in parallel, however the communication among the subpopulations emphasizes population fitness values but ignores the population diversity. The proposed ANDE-NNE adapts the population by considering the accuracy and diversity, which facilitates a strong NNE. Furthermore, Fig. 6 gives a more intuitive view of the percentage of improvements made by ANDE-NNE against DE-NNE, BP-NNE, and MCCS, respectively. As we can observe from the figure that ANDE-NNE performs the most outstanding results and the range of improvements gained by ANDE-NNE is excellent in most situations. The above observation validates the competitive performance of ANDE-NNE.

Table 8 summarizes the average consumed time of different NNE methods. In the table, BP-NNE costs 0.72s, which is the fastest. For the EA-based methods, ANDE-NNE averagely consumes 3.47s, while the other two methods, i.e., DE-NNE and MCCS, take around 20 times longer than ANDE-NNE. This is because ANDE-NNE optimizes all NNs in a single execution while the other comparison methods require repeated executions to train each individual NN.

5.4. Further analysis of ANDE-NNE

5.4.1. Investigation of NN size obtained by ANDE-NNE

The average number of individual NNs obtained by ANDE-NNE on all datasets is illustrated in Fig. 7. First, it can be observed that, in different cases, the number of NNs in ANDE-NNE is quite different. ANDE-NNE adjusts the number of NNs in the training for different datasets. The NNs with unsuitable number may weaken the overall effect for ensemble, as the excessive number of NNs will bring redundancy and the insufficient number of NNs can not provide enough generalization ability of voters. For example, the number of NNs produced by ANDE-NNE for Ionosphere dataset is much fewer than that of DE-NNE, BP-NNE, and MCCS, while the performance of ANDE-NNE is much better than that of the rival NNEs whose NN size is fixed at 20. Second, even when the number of the generated NNs is close to 20 (the fixed NN number adopted for the comparison methods), ANDE-NNE shows a more promising performance than the others. For example, the number of NNs produced by ANDE-NNE for Iris dataset is close to 20, and in this case ANDE-NNE still performs better than the other methods with 20 fixed NNs (see Fig. 6). This owes much to the powerful multimodal optimization ability of ANDE, which guarantees the good accuracy and diversity of the produced NNs.

5.4.2. Investigation of NN topology of ANDE-NNE

The weak NNs are assembled to yield a strong NNE. The topology of base NNs affects the performance NNE. We investigate the effect among various NN topologies: a hidden layer with three neurons, a hidden layer with five neurons, a hidden layer with ten neurons, two hidden layers with three neurons each, and two hidden layers with five neurons each. Table 9 reports the validation accuracy of the five different topologies and their significance test results. The significance is summarized with a triplet in the form of "$a|b|c$", where *"a"* denotes the number of cases that the ANDE-NNE with default setting (i.e., one hidden layer with five neurons) significantly outperforms the competitor, *"b"* indicates the number of cases that the difference of the results between the compared algorithms is insignificant, and the last *"c"* suggests the number of cases that the default ANDE-NNE performs significantly worse than the other. From the table, we can see that ANDE-NNE variants have similar validation accuracy values. The results indicate that, owing to the good generalization ability of ensemble learning, ANDE-NNE is not very sensitive to the topologies of individual NNs. As the

Table 7
Validation accuracy measures on the different datasets (average and standard deviation).

	ANDE-NNE		BP-NN		BP-NNE		DE-NN		DE-NNE		MCCS	
	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
Haberman	**77.10**	5.98	70.97 (+)	5.89	69.68 (+)	6.31	73.55 (≈)	4.51	70.97 (≈)	6.97	71.94 (≈)	8.74
Heart	**82.22**	9.21	71.85 (≈)	16.02	74.44 (≈)	7.29	77.78 (≈)	9.07	80.37 (≈)	8.74	87.04 (≈)	6.36
Diabetes	**76.75**	4.08	76.10 (≈)	4.55	75.84 (≈)	5.55	66.62 (+)	4.74	68.70 (+)	5.14	71.69 (+)	4.69
Ionosphere	**90.56**	5.59	71.67 (+)	8.86	74.44 (+)	14.57	63.89 (+)	6.55	72.50 (+)	9.39	83.33 (+)	6.28
Wisconsin	97.39	2.25	69.71 (+)	10.92	85.22 (+)	15.12	97.10 (≈)	2.05	**97.83** (≈)	1.71	97.54 (≈)	2.06
Wine	**95.56**	7.31	92.78 (≈)	8.71	95.56 (≈)	6.83	63.89 (+)	15.77	79.44 (+)	20.80	82.22 (+)	16.10
Balance	**90.16**	4.78	87.94 (≈)	4.04	88.89 (≈)	4.43	86.98 (≈)	2.97	86.35 (≈)	4.31	89.52 (≈)	4.56
Hayes-roth	**70.00**	12.05	66.43 (≈)	8.28	69.29 (≈)	11.19	47.86 (+)	15.81	50.00 (+)	15.79	50.71 (+)	17.96
Iris	**97.33**	3.44	94.00 (≈)	8.58	96.00 (≈)	7.17	71.33 (+)	14.07	77.33 (+)	14.47	71.33 (+)	19.89
Seeds	**96.19**	3.76	42.86 (+)	20.20	90.48 (≈)	12.30	82.86 (+)	9.04	73.81 (+)	15.75	91.43 (+)	3.76
Car	**83.18**	1.53	71.79 (+)	3.26	67.46 (+)	3.36	76.53 (+)	4.07	71.56 (+)	3.65	77.63 (+)	2.23
Zoo	81.82	10.50	85.45 (≈)	10.67	**88.18** (≈)	7.48	56.36 (+)	16.49	46.36 (+)	15.72	70.91 (+)	9.39
Overall	**86.52**	5.87	75.13	9.16	81.29	8.47	72.06	8.76	72.94	10.20	78.77	8.50

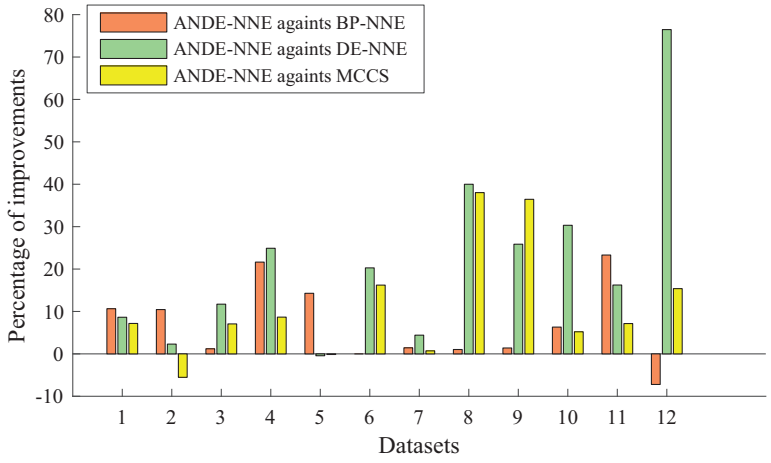


Fig. 6. The percentage of improvements made by ANDE-NNE against DE-NNE, BP-NNE, and MCCS (figures in the horizontal axis demonstrate the different datasets: 1- Haberman, 2- Heart, 3- Diabetes, 4- Ionosphere, 5- Wisconsin, 6- Wine, 7- Balance, 8- Hayes-roth, 9- Iris, 10- Seeds, 11- Car, 12- Zoo). ANDE-NNE shows improvement over other NNE optimization methods on most datasets.

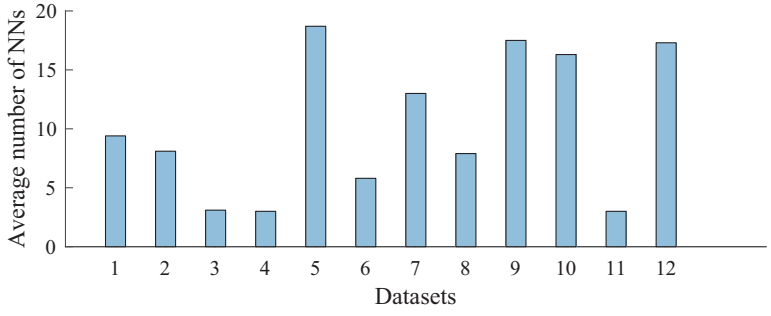


Fig. 7. The average number of individual NNs used in ANDE-NNE (figures in the horizontal axis demonstrate the different datasets: 1- Haberman, 2- Heart, 3- Diabetes, 4- Ionosphere, 5- Wisconsin, 6- Wine, 7- Balance, 8- Hayes-roth, 9- Iris, 10- Seeds, 11- Car, 12- Zoo). ANDE-NNE applies a flexible number of NNs for different datasets.

Table 8
Average execution time of NNE methods.

Algorithm	ANDE-NNE	BP-NNE	DE-NNE	MCCS
Time (s)	3.47	0.72	61.31	56.16

Table 9
Validation accuracy of ANDE-NNE with different topologies.

Hidden layer	(3)	(5)	(10)	(3, 3)	(5, 5)
Validation accuracy	88.01	87.98	88.19	85.25	86.51
Validation significance	1/11/0	–	0/12/0	7/5/0	4/8/0

simple NN topology endows ANDE-NNE with sufficient capability to vote in NNE, we suggest simple NN topology, i.e., one hidden layer with three to ten hidden neurons.

In practice, a complex NN is more preferred when only one NN is utilized to accomplish the machine learning task. On the contrary, the ensemble learning integrates multiple “simple/weak” base learners to obtain a strong ensemble learner. In the ensemble learning, diverse base learners enable to reduce the dependency on data distribution and enhance the generalization capability of the ensemble learner. In this aspect, adopting base learners with complex structure is contrary to the original intention of learning en-

semble. Therefore, the simple NN is generally adopted in ensemble learning.

6. Conclusion

This paper develops an adaptive niching differential evolution algorithm named ANDE for tackling MMOPs and training NNEs. Three novel strategies are proposed to enhance the performance of the algorithm: (1) the heuristic clustering method that divides the population into non-overlapping subpopulations; (2) the dynamic population adaptation mechanism that contributes to improving diversity for the stagnating population and saving the unnecessary computational budget by removing the overlapping subpopulations, and F and CR adaptation mechanism for fine-tuning the parameters along the evolution; (3) the auxiliary movement scheme for an equilibrium between exploration and exploitation.

Based on these novel techniques, the experimental results of ANDE have shown the powerfulness. The first experiment part is conducted on 20 multimodal test functions of the CEC2013 benchmark suite. ANDE outperforms the state-of-the-art niching EAs in terms of PR, SR, and CS. Besides, we carry out an investigation of effectiveness of ANDE's components. The components of ANDE play different roles and together contribute a powerful optimization ability. Subsequently, in part two, ANDE is adopted to train NNE, that is ANDE-NNE, on a twelve classification instances of UCI repository. ANDE-NNE shows promising performance when compared with other five NNE training methods in terms of validation accuracy measure. Furthermore, we provide a deep investigation of the obtained NN size. The results indicate that ANDE-NNE offers tailored NN size on the specific dataset.

In this study, the concurrent optimization of multiple NNs considers the weight parameters only. In the future, it would be interesting to encode the hyperparameters, such as the network topology, for optimization. In addition, the base NNs are homogeneous in the ANDE-NNE. It remains a challenging task to train an ensemble with heterogeneous base learners in a concurrent way. The proposed method can also bring benefits to a wide range of applications that require the diversity of models, such as developing ensemble methods to tackle the concept drift problem in an incremental learning environment

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Ting Huang: Methodology, Software, Validation, Investigation, Writing - review & editing. **Dan-Ting Duan:** Methodology, Software, Validation, Investigation, Writing - original draft. **Yue-Jiao Gong:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Funding acquisition. **Long Ye:** Writing - review & editing. **Wing W.Y. Ng:** Writing - review & editing. **Jun Zhang:** Writing - review & editing.

References

[1] M. Perc, M. Ozer, J. Hojnik, Social and juristic challenges of artificial intelligence, Palgrave Commun. 5 (1) (2019) 61.
[2] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 993–1001.
[3] Y.-T. Yan, Y.-P. Zhang, Y.-W. Zhang, X.-Q. Du, A selective neural network ensemble classification for incomplete data, Int. J. Mach. Learn. Cybern. 8 (5) (2017) 1513–1524.
[4] F.-Y. Xie, H.-D. Fan, Y. Li, Z.-G. Jiang, R.-S. Meng, A.C. Bovik, Melanoma classification on dermoscopy images using a neural network ensemble model, IEEE Trans. Med. Imag. 36 (2017) 849–858.
[5] Z. Long, G.-X. Yu, D.-W. Xia, J. Wang, Protein-protein interactions prediction based on ensemble deep neural networks, Neurocomputing 324 (2018) 10–19.
[6] C.-X. Ding, D.-C. Tao, Trunk-branch ensemble convolutional neural networks for video-based face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2018) 1002–1014.
[7] S. Li, P. Wang, L. Goel, Wind power forecasting using neural network ensembles with feature selection, IEEE Trans. Sustain. Energy 6 (2015) 1447–1456.
[8] J. Jin, K. Fu, C. Zhang, Traffic sign recognition with hinge loss trained convolutional neural networks, IEEE Trans. Intell. Transp. Syst. 15 (2014) 1991–2000.
[9] S. Venkatraman, S. Kulkarni, MapReduce neural network framework for efficient content based image retrieval from large datasets in the cloud, in: Proceedings of the 12th International Conference on Hybrid Intelligent Systems, 2012, pp. 63–68.
[10] D. Helbing, D. Brockmann, T. Chadefaux, K. Donnay, U. Blanke, O. Woolley-Meza, M. Moussaid, A. Johansson, J. Krause, S. Schutte, M. Perc, Saving human lives: What complexity science and information systems can contribute, J. Stat. Phys. 158 (3) (2015) 735–781.
[11] J. Yang, X. Zeng, S. Zhong, S. Wu, Effective neural network ensemble approach for improving generalization performance, IEEE Trans. Neural Netw. Learn. Syst. 24 (2013) 878–887.
[12] H.R. Bonab, F. Can, Less is more: a comprehensive framework for the number of components of ensemble classifiers, IEEE Trans. Neural Netw. Learn. Syst. 30 (2017) 2735–2745.
[13] Y.-H. Jia, W.-N. Chen, T.-L. Gu, H.-X. Zhang, H.-Q. Yuan, S. Kwong, J. Zhang, Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization, IEEE Trans. Evol. Comput. 23 (2019) 188–202.
[14] Y.-H. Zhang, Y.-J. Gong, T.-L. Gu, H.-Q. Yuan, W. Zhang, S. Kwong, J. Zhang, DECAL: decomposition-based coevolutionary algorithm for many-objective optimization, IEEE Trans. Cybern. 49 (2017) 27–41.
[15] Y.-H. Zhang, Y.-J. Gong, W.-N. Chen, T.-L. Gu, H.-Q. Yuan, J. Zhang, A dual-colony ant algorithm for the receiving and shipping door assignments in cross-docks, IEEE Trans. Intell. Transp. Syst. 20 (2019) 2523–2539.
[16] Y.-J. Gong, E. Chen, X.-L. Zhang, L.M. Ni, J. Zhang, Antmapper: an ant colony-based map matching approach for trajectory-based applications, IEEE Trans. Intell. Transp. Syst. 19 (2018) 390–401.
[17] Z.-G. Chen, Z.-H. Zhan, Y. Lin, Y.-J. Gong, T.-L. Gu, F. Zhao, H.-Q. Yuan, X.-F. Chen, Q. Li, J. Zhang, Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach, IEEE Trans. Cybern. 49 (2019) 2912–2926.
[18] T. Huang, Y.-J. Gong, S. Kwong, H. Wang, J. Zhang, A niching memetic algorithm for multi-solution traveling salesman problem, IEEE Trans. Evol. Comput. (2019). 1–1
[19] I. Fister, M. Perc, S.M. Kamal, I. Fister, A review of chaos-based firefly algorithms: perspectives and research challenges, Appl. Math. Comput. 252 (2015) 155–165.
[20] T. Huang, Y. Gong, Y. Zhang, Z. Zhan, J. Zhang, Automatic planning of multiple itineraries: A niching genetic evolution approach, IEEE Trans. Intell. Transp. Syst. (2019) 1–16.
[21] Y.-K. Kwon, B.R. Moon, A hybrid neurogenetic approach for stock forecasting, IEEE Trans. Neural Netw. 18 (2007) 851–864.
[22] Z. Zhao, X. Feng, Y. Lin, F. Wei, S. Wang, T. Xiao, M. Cao, Z. Hou, Evolved neural network ensemble by multiple heterogeneous swarm intelligence, Neurocomputing 149 (2015) 29–38.
[23] W. Sheng, P. Shan, S. Chen, Y. Liu, F.E. Alsaadi, A niching evolutionary algorithm with adaptive negative correlation learning for neural network ensemble, Neurocomputing 247 (2017) 173–182.
[24] R. Thomsen, Multimodal optimization using crowding-based differential evolution, in: Proceedings of the 2004 Congress on Evolutionary Computation, 2004, pp. 1382–1389.
[25] E. Dilettoso, N. Salerno, A self-adaptive niching genetic algorithm for multimodal optimization of electromagnetic devices, IEEE Trans. Magn. 42 (2006) 1203–1206.
[26] X. Li, Efficient differential evolution using speciation for multimodal function optimization, in: Proceedings of the Seventh Annual Conference on Genetic and Evolutionary Computation, 2005, pp. 873–880.
[27] A. Pétrowski, A clearing procedure as a niching method for genetic algorithms, in: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 798–803.
[28] X.-D. Yin, N. Germay, A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization, in: Artificial Neural Nets and Genetic Algorithms, Springer, 1993, pp. 450–457.
[29] G.R. Harik, Finding multimodal solutions using restricted tournament selection, in: Proceedings of the 6th International Conference on Genetic Algorithms, 1995, pp. 24–31.
[30] Y.-J. Gong, J. Zhang, Y.-C. Zhou, Learning multimodal parameters: a bare-bones niching differential evolution approach, IEEE Trans. Neural Netw. Learn. Syst. 29 (2018) 2944–2959.
[31] S. Biswas, S. Kundu, S. Das, Inducing niching behavior in differential evolution through local information sharing, IEEE Trans. Evol. Comput. 19 (2015) 246–263.

[32] Z. Wang, Z. Zhan, Y. Lin, W. Yu, H. Yuan, T. Gu, S. Kwong, J. Zhang, Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems, IEEE Trans. Evol. Comput. 22 (2018) 894–908.
[33] B. Qu, P.N. Suganthan, J. Liang, Differential evolution with neighborhood mutation for multimodal optimization, IEEE Trans. Evol. Comput. 16 (2012) 601–614.
[34] M.G. Epitropakis, X. Li, E.K. Burke, A dynamic archive niching differential evolution algorithm for multimodal optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2013, pp. 79–86.
[35] W. Gao, G.G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE Trans. Cybern. 44 (2014) 1314–1327.
[36] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution, IEEE Trans. Cybern. 44 (2014) 1726–1737.
[37] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (2006) 646–657.
[38] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417.
[39] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958.
[40] R. Storn, K.V. Price, Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[41] P. Civicioglu, E. Besdok, Bernstain-search differential evolution algorithm for numerical function optimization, Expert Syst. Appl. 138 (2019) 112831.
[42] B. Xu, H. Zhang, M. Zhang, L. Liu, Differential evolution using cooperative ranking-based mutation operators for constrained optimization, Swarm Evol. Comput. 49 (2019) 206–219.
[43] L. Peng, S. Liu, R. Liu, L. Wang, Effective long short-term memory with differential evolution algorithm for electricity price prediction, Energy 162 (2018) 1301–1314.
[44] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, P. Suganthan, Ensemble of differential evolution variants, Inf. Sci. 423 (2018) 172–186.
[45] L. Wang, H. Hu, X.-Y. Ai, H. Liu, Effective electricity energy consumption forecasting using echo state network improved by differential evolution algorithm, Energy 153 (2018) 801–815.
[46] S. Suganthi, D. Devaraj, K. Ramar, S.H. Thilagar, An improved differential evolution algorithm for congestion management in the presence of wind turbine generators, Renew. Sustain. Energy Rev. 81 (2018) 635–642.
[47] W.S. Sakr, R.A. EL-Sehiemy, A.M. Azmy, Adaptive differential evolution algorithm for efficient reactive power management, Appl. Soft Comput. 53 (2017) 336–351.
[48] Q. Yang, W.-N. Chen, Z.-T. Yu, T.-L. Gu, Y. Li, H.-X. Zhang, J. Zhang, Adaptive multimodal continuous ant colony optimization, IEEE Trans. Evol. Comput. 21 (2017) 191–205.
[49] Y.-H. Zhang, Y.-J. Gong, H.-X. Zhang, T.-L. Gu, Z. Jun, Towards fast niching evolutionary algorithms: a locality sensitive hashing-based approach, IEEE Trans. Evol. Comput. 21 (2017) 347–362.
[50] X. Li, A multimodal particle swarm optimizer based on fitness euclidean-distance ratio, in: Proceedings of the Ninth Annual Conference on Genetic and Evolutionary Computation, 2007, pp. 78–85.
[51] X.-D. Li, Niching without niching parameters: particle swarm optimization using a ring topology, IEEE Trans. Evol. Comput. 14 (2010) 150–169.
[52] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66.
[53] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (2011) 4–31.
[54] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Multimodal optimization using niching differential evolution with index-based neighborhoods, in: 2012 IEEE Congress on Evolutionary Computation, 2012, pp. 1–8.
[55] B. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evol. Comput. 17 (2013) 387–402.
[56] Q. Yang, W.-N. Chen, Y. Li, C.L.P. Chen, X.-M. Xu, J. Zhang, Multimodal estimation of distribution algorithms, IEEE Trans. Cybern. 47 (2017) 636–650.
[57] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (1990) 197–227.
[58] Y. Freund, Boosting a weak learning algorithm by majority, Inf. Comput. 121 (2) (1995) 256–285.
[59] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[60] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[61] F. Moretti, S. Pizzuti, S. Panzieri, M. Annunziato, Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling, Neurocomputing 167 (2015) 3–7.
[62] D. Rumelhart, J. Mcclelland, Learning internal representations by error propagation, Read. Cogn. Sci. 323 (6088) (1988) 399–421.
[63] G. Liao, T. Tsao, Application of a fuzzy neural network combined with a chaos genetic algorithm and simulated annealing to short-term load forecasting, IEEE Trans. Evol. Comput. 10 (2006) 330–340.
[64] H.D. Chiang, L.G. Chen, R.P. Liu, N. Dong, Group-based chaos genetic algorithm and non-linear ensemble of neural networks for short-term load forecasting, IET Gen. Transm. Distrib. 10 (2016) 1440–1447.
[65] C. Castillo, G. Nitschke, A.P. Engelbrecht, Niche particle swarm optimization for neural network ensembles, in: Advances in Artificial Life. Darwin Meets von Neumann, 5778, 2009, pp. 399–407.
[66] R. Brits, A.P. Engelbrecht, F. Van den Bergh, A niching particle swarm optimizer, in: Proceedings of the 4th Asia-pacific Conference on Simulated Evolution and Learning, 2, 2002, pp. 692–696.
[67] R.K. Ursem, Multinational gas: Multimodal optimization techniques in dynamic environments, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2000, pp. 19–26.
[68] R. Mendes, A.S. Mohais, DynDE: A differential evolution for dynamic optimization problems, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2005, pp. 2808–2815.
[69] X.-D. Li, A. Engelbrecht, M.G. Epitropakis, Benchmark functions for CEC2013 special session and competition on niching methods for multimodal function optimization, RMIT University, Evolutionary Computation and Machine Learning Group, Australia, 2013. Technical Report
[70] J. Yao, N.N. Kharma, P. Grogono, Bi-objective multipopulation genetic algorithm for multimodal function optimization, IEEE Trans. Evol. Comput. 14 (2010) 80–102.
[71] Y. Wang, H.-X. Li, G.G. Yen, W. Song, MOMMOP: multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems, IEEE Trans. Cybern. 45 (2015) 830–843.
[72] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.
[73] D. Dua, C. Graff, UCI machine learning repository, 2017, http://archive.ics.uci.edu/ml.

Ting Huang is currently pursuing the Ph.D. degree in School of Computer Science and Engineering, South China University of Technology, China. Her current research interests include evolutionary computation, swarm intelligence, multi-solution optimization, and their real-world applications.

Dan-Ting Duan received the B.E. degree in Computer Science from Chongqing University, Chongqing, China, in 2016. She is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, Communication University of China, Beijing, China. Her current research interests include evolutionary computation algorithms, facial expression (micro-expression) recognition, and multimodal optimization algorithms.

Yue-Jiao Gong received the B.S. and Ph.D. degree in Computer Science from Sun Yat-sen University, China, in 2010 and 2014, respectively. Currently, she is a Full Professor with the School of Computer Science and Engineering, South China University of Technology, China. Her research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation and smart city scheduling. She has published over 80 papers, including more than 30 IEEE Transactions papers, in her research area.

Long Ye received the B.Eng. degree in electronic engineering from Shandong University, Jinan, China, in 2003, the M.Eng. degree and Dr.Eng. degree from Communication University of China, Beijing, China, in 2006 and 2012, respectively. He is currently with the Key Laboratory of Media Audio & Video of Communication University of China of Ministry of Education. Prior to that, he was a visiting scholar at Ryerson University, Toronto, Canada. His research interests include computer vision, Image Compression and Virtual Reality.

Wing W. Y. Ng received the B.Sc. and Ph.D. degrees in computer science from Hong Kong Polytechnic University, Hong Kong, in 2001 and 2006, respectively. He is a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. He is currently the Deputy Director of the Guangdong Provincial Key Laboratory of the Computational Intelligence and Cyberspace Information Guangzhou, China. He is the Principle Investigator of four China National Nature Science Foundation Projects and a Program for New Century Excellent Talents in University from the Ministry of Education, China. His research interests include neural networks, deep learning, smart grid, smart health care, smart manufacturing, and non stationary information retrieval. Dr. Ng is an Associate Editor of the International Journal of Machine Learning and Cybernetics. He served as the Board of Governor of IEEE Systems, Man and Cybernetics Society in 2011-2013.

Jun Zhang received the Ph.D. degree from the City University of Hong Kong, Kowloon, Hong Kong, in 2002. He is currently a visiting scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits. Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE Transactions on Evolutionary Computation, the IEEE Transactions on Cybernetics, and the IEEE Transactions on Industrial Electronics.