Learning Multimodal Parameters: A Bare-Bones Niching Differential Evolution Approach

Yue-Jiao Gong, *Member, IEEE*, Jun Zhang, *Fellow, IEEE*, and Yicong Zhou, *Senior Member, IEEE*

***Abstract*— Most learning methods contain optimization as a substep, where the nondifferentiability and multimodality of objectives push forward the interplay of evolutionary optimization algorithms and machine learning models. The recently emerged evolutionary multimodal optimization (MMOP) technique enables the learning of diverse sets of effective parameters for the models simultaneously, providing new opportunities to the applications requiring both accuracy and diversity, such as ensemble, interactive, and interpretive learning. Targeting at locating multiple optima simultaneously in the multimodal landscape, this paper develops an efficient neighborhood-based niching algorithm. Bare-bones differential evolution is used as the baseline. Further, using Gaussian mutation with local mean and standard deviations, the neighborhoods capture niches that match well with the contours of peaks in the landscape. To increase diversity and enhance global exploration, the proposed algorithm embeds a diversity preserving operator to reinitialize converged or overlapped neighborhoods. The experimental results verify that the proposed algorithm has superior and consistent performance for a wide range of MMOP problems. Further, the algorithm has been successfully applied to train neural network ensembles, which validates its effectiveness and benefits of learning multimodal parameters.**

***Index Terms*— Fitness landscape, Gaussian model, multimodal optimization (MMOP), neighborhood strategy, neural network ensemble (NNE), niching.**

I. Introduction

MANY machine learning methods involve the task of searching parameters to optimize the objective functions, such as training the parameters of neural networks and kernel methods. Considering the training procedure as solving the optimization problem, it is commonly recognized that the objective function usually exhibits some unwelcome properties including nondifferentiability and multimodality. These properties challenge the traditional gradient methods and give birth to the use of alternatives, e.g., the evolutionary computation-based optimizers [1]–[5]. Evolutionary algorithms (EAs) have been widely adopted to solve parameter optimization problem, owing much to the following two advantages. First, EAs have wide availability since they do not require the objective functions to satisfy some mathematically sound properties like differentiability and convexity. Thus, the use of EAs poses no restriction on formulating the objectives. Second, EAs are able to locate globally optimal results, which are insensitive to the initial distribution of parameters. Thus, their performance is persistent under different problem landscapes.

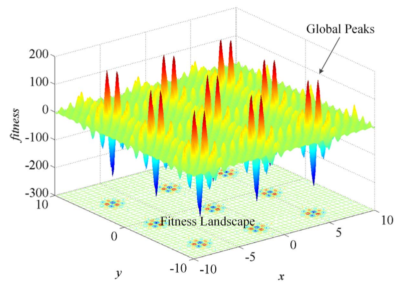


Fig. 1. Shubert 2D problem space contains 18 unevenly distributed global peaks. The peaks can be further divided into nine clusters, each containing two closely located peaks. Besides, there are a huge number of local optima in this landscape.

To optimize the parameters of machine learning models, traditionally, the EAs focus on searching for a single global optimum. Nevertheless, many problem spaces contain a set of optimal solutions possessing similar or equal fitness values, as can be observed in Fig. 1. These solutions typically scatter in different positions of the problem space. Learning multiple sets of effective parameters simultaneously has wide applications such as 1) generating diverse and accurate base learners for ensemble learning [6]; 2) introducing posterior user preferences [7]; 3) providing geometric interpretation for the problem distribution [8]; and 4) tackling physical constraints and system dynamics [9]. The task can be completed by the recently emerged multimodal optimization (MMOP) technique [10]–[13].

EAs cannot be directly used for MMOP because their global selection and recombination operators make the entire population converge to a single position. Considering this, niching techniques have been widely incorporated into EAs for the sake of enhancing diversity and maintaining multiple optima (within different niches). The related methods include clustering [11], classification [12], ensemble method [13],

Manuscript received May 30, 2016; revised December 15, 2016 and April 6, 2017; accepted May 22, 2017. Date of publication June 20, 2017; date of current version June 21, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61502542 and Grant 61332002, in part by the Macau Science and Technology Development Fund under Grant FDCT/016/2015/A1, and in part by the Research Committee at University of Macau under Grant MYRG2014-00003-FST and Grant MYRG2016-00123-FST. *(Corresponding author: Yicong Zhou.)*

Y.-J. Gong is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: gongyuejiao@gmail.com).

J. Zhang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: junzhang@ieee.org).

Y. Zhou is with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: yicongzhou@umac.mo).

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

Digital Object Identifier 10.1109/TNNLS.2017.2708712

crowding [14], fitness sharing [9], speciation [15], neighborhood strategies [16], and multiobjective search [17]. These niching methods have successfully enabled EAs for MMOP while some critical issues remain to be resolved. First, many existing niching techniques introduce new parameters that are very sensitive to the problem landscapes. In order to improve the niching performance, these parameters need to be fine-tuned on different problems, which has become a bottleneck that hinders the wide applicability of niching algorithms. Second, some algorithms suffer from high computational costs on computing and sorting the distance between solutions in the population. Although obtaining improved results, the required optimization time is long. These algorithms are hence not suitable for solving the time-critical real-world problems. Third, for many problems whose landscapes are complex (e.g., highly correlated variables, unevenly distributed global optima, and numerous local optima), existing algorithms may fail to provide satisfactory results.

This paper develops a novel niching algorithm to address the above-mentioned issues in learning multimodal parameters. We adopt the Gaussian bare-bones differential evolution (GBDE) [18] as the baseline optimization algorithm. GBDE belongs to the bare-bones EA family, a branch of EAs that has a simple and almost parameter-free framework and achieves competitive performance [19]–[22]. For the purpose of locating multimodal optima, we incorporate an index-based neighborhood strategy, together with a diversity-preserving operation (DPO), into the algorithm to realize niching. Compared with existing work, the novelties and advantages of our method are summarized as follows.

1) *Bare-Bones Model:* To the best of our knowledge, we make the first attempt to solve MMOP using a bare-bones EA variant. The use of bare-bones baseline model alleviates the burden of fine-tuning the basic control parameters of EAs. In addition, our method is free from fine-tuning the niche radius, a parameter that is very sensitive to the problem landscape. Although Bare-Bones Niching DE (BNDE) involves a few parameters, these parameters are either fixed or adaptively adjusted by the algorithm. In other words, we no longer need to manually investigate the parameters for different problems as traditionally did.
2) *Neighborhood Niching Strategy:* The neighborhood niching strategy involves no extra computational cost, making the proposed algorithm cost efficient. Compared with the state-of-the-art competitive algorithms, our algorithm requires much less computational time. Based on the neighborhood strategy, we specifically design a Gaussian mutation to enhance local exploitation of different peaks.
3) *Diversity-Preserving Operation:* One feature of the proposed approach is that we first design the DPO to reinitialize converged or overlapped niches. In the scenario of MMOP, the requirement of diversity is very critical since the multiple optima can be located far from each other. Meanwhile, as the algorithm needs to approach several peaks simultaneously, the average computational resources available for exploiting each peak are very limited. The proposed DPO not only maintains diversity, but also saves inefficient search efforts. Using the DPO together with the local Gaussian mutation improves both exploitation and exploration.

Experiments indicate that the proposed algorithm generally outperforms 12 state-of-the-art niching algorithms on standard MMOP benchmarks. Further, we apply the BNDE algorithm to train a set of neural networks for neural network ensembles (BNDE-NNE). The results validate the promising performance of BNDE-NNE.

The rest of this paper is organized as follows. Section II presents the background and related work. Section III describes the GBDE. Section IV presents details of the proposed niching method; experiments are conducted in Section V; further, in Section VI, the algorithm is applied to learn multimodal parameters of NNEs; finally, conclusions are drawn in Section VII.

II. Background and Related Work

A. Evolutionary Optimization Meets Machine Learning

The interaction of evolutionary optimization and machine learning is witnessed to play a crucial role in the area of modern computational intelligence.

On the one hand, most machine learning methods involve the task of finding parameters to minimize the cost functions, where optimization is proven to be of vital importance in the performance. Many studies exploited using EAs for improving the performance of machine learning, among which the most well-known application is probably the training of neural network models. Training the structural parameters of neural network is known to be a nondifferentiable and multimodal problem, making the gradient descent-based optimization methods such as the backpropagation (BP) and least squares unavailable or unstable. Recently, owing to the global optimization ability and good robustness, EAs have received attention in optimizing the structure and connecting weights of different neural networks, such as the autoencoders [3], multilayer perceptron neural networks [1], [23], deep belief networks [24], [25], and fuzzy neural networks [2], [26]. In addition, EAs have been widely adopted for kernel prediction in various kernel-involved methods including radial basis function neural networks [27], [28], support vector machines [29], support vector regression [4], and kernel principle component analysis [5]. For example, Han *et al.* [27] developed a very comprehensive approach that applies an adaptive particle swarm optimization algorithm to optimize the center and width of Gaussian kernel for each neuron, the output weights of neurons, and the network size, simultaneously. Results indicated that the algorithm outperforms the others in solving nonlinear learning problems.

On the other hand, with the rapid development of machine learning, the area is no longer a consumer of optimization, but itself helps enhance evolutionary computation [30] or derive new optimization algorithms [31], [32]. Considering the context of MMOP, Dong and Zhou [12] developed a Gaussian classifier-based evolutionary strategy that tackles MMOP problems as classification problems and employs the Gaussian mixture models to identify the optima.

B. Learning Multimodal Parameters

Traditionally, when applying EAs to optimize parameters of machine learning models, a single globally optimal solution is provided. With the advent of niching EAs, it is now possible to achieve a diverse set of similarly fit optima for the multimodal space, providing opportunities of learning multiple sets of effective parameters simultaneously (named learning multimodal parameters). In this section, we discuss the advantages and potential applications of this new technique.

First, learning multimodal parameters assists in generating base learners in ensemble learning. Given multiple sets of parameter settings, a number of base learners (e.g., neural networks [33]) are generated to complete the same task. Owing to the characteristics of MMOP, the base learners possess good accuracy and diversity, which meet the two essential requirements of ensemble learning. Combining the base learners together reduces the bias and variance terms in the prediction error of the entire system.

Second, learning multimodal parameters provide opportunities of introducing user preferences in learning. In many applications, posterior user preferences are very important. It is required to provide as many solutions as possible, so that the user can select the most proper one based on his domain knowledge. This requirement is commonly existed in the field of feature matching and pattern recognition. For example, in [7] and [34], the MMOP techniques were used to generate multiple effective feature sets simultaneously. In addition to the benefits of providing selective choices for posterior preferences, the study in [35] indicated that the niching results could show the dependence of different groups of features. Besides, another possible application field is the interactive machine learning: the system outputs a set of results for humans to evaluate, select, and correct [36].

Third, the distribution of multimodal parameters helps to reveal some insights of the problem at hand, which are useful to discover the hidden key properties of some substances. For example, the accurate distribution of minima provides a geometric interpretation of the interacting boson model (IBM-2) on energy, exited states, and nuclear shape-phase transitions [8].

Forth, learning multimodal parameters provide a way to deal with physical constraints or system dynamics. Many practical learning systems encounter physical constraints or dynamics that can hardly be incorporated into the model. If the algorithm searches for a single optimum, the output solution may be impractical to be realized due to the physical limitation, or the solution may have a lifespan that it becomes less effective after a time period. In both cases, a set of equally fit solutions is desirable, as it provides alternatives for keeping the system optimality. A typical example is the design of electromagnetic devices, to which a niching genetic algorithm shows its effectiveness [9].

Besides the above benefits of learning multimodal parameters, the MMOP itself can be transferred to solve clustering problems by considering the location of each optimum as the sentinel of a cluster [37], [38]. The MMOP clustering technique has advantages over traditional clustering methods (like k-means) in the global optimization ability and the self-adjustment of the cluster number.

C. Niching for Multimodal Optimization

The concept of niching is inspired by the evolution process of organisms in natural systems. Instead of merging into a single population, a number of subpopulations (or species) evolve separately and interactively to fit themselves into different subspaces of the environment. In terms of MMOP, niching refers to maintaining different subpopulations during the whole evolution process in order to locate multiple optima. This section gives a brief review of several representative niching techniques.

The most commonly used niching techniques are crowding and speciation. Crowding operates on the selection or individual replacement process to avoid the competition between dissimilar individuals [14]. For each offspring, it is compared with the nearest individual from crowding factor (*CF*) randomly sampled individuals. If the offspring has a higher fitness value, it replaces the selected nearest individual. A small *CF* may lead to replacement error that the offspring replaces a dissimilar individual, resulting in poor population diversity. Thus, it is suggested to set *CF* to a large value or equal to the population size. Speciation classifies individuals in the population into different species, each dominated by a species seed with the best fitness value [15]. Considering the seed as the species center, a parameter r_s further defines the radius of the species. Then, the evolution operators of EAs are conducted in each species separately. Many niching algorithms in the literature used speciation. A difficulty of speciation-based EAs is that the proper setting of the species radius r_s for different problems is hard to know in advance.

To alleviate the burden of setting the radius of niches or species, several neighborhood-based niching strategies were developed in recent years. Li [39] showed that a local version of particle swarm optimization (PSO), such as the PSO with ring topology, is able to tackle MMOP. Later, Epitropakis *et al.* [40] incorporated both ring and von Neumann neighborhoods into the DE/nrand algorithms. Besides eliminating the radius parameter, these index-based neighborhood strategies have another advantage that they reduce the computational cost of formulating the niches or species. On the other hand, Qu *et al.* [16], [41] pointed out that, compared with index-based neighborhood strategies, distance-based methods perform better in maintaining diversity and forming different niches. In their proposed locally informed particle swarm (LIPS) [16] and neighborhood-based DE family [neighborhood-based crowding DE (NCDE), neighborhood-based speciation DE (NSDE), and Neighborhood-based sharing DE (NShDE)] [41], each individual performs evolution operation in the neighborhood formed by its n nearest individuals. In the parent-centric normalized mutation DE (PNPCDE) [42], the neighborhood is built by a distance-based probabilistic selection model. Further, the locally informative speciation DE (LoISDE) [43] combines distance-based neighborhood and speciation strategies. Although inducing extra computational cost of computing and sorting the distance, the distance-based neighborhood strategies have shown their promising performance in locating multiple optima.

Apart from the above-reviewed work, there are some other methods such as fitness sharing [9], clustering [11], classification [12], ensemble method [13], and multiobjective search [17]. However, as described in Section I, many of these works are either relying on problem-dependent parameters or computationally expensive. Moreover, some algorithms fail to provide satisfactory results when the number of optima in the problem increases. This paper proposes a novel niching algorithm named BNDE to alleviate such problems.

III. Gaussian Bare-Bones Differential Evolution

GBDE is a variant of DE using the Gaussian sampling mutation [18], and it belongs to the bare-bones EA family [19]–[22]. In the initialization, a population of individuals $\boldsymbol{P} = \{X_{i,0} = [x_{i,1,0}, x_{i,2,0}, \ldots, x_{i,D,0}] \mid i = 1, 2, \ldots, N\}$ are randomly distributed in the D-dimensional problem space. Then, the algorithm enters a loop that, at each generation G, mutation, crossover, and selection operators are performed to update the population. In the Gaussian sampling mutation of GBDE, the mutant vector $V_{i,G} = [v_{i,1,G}, v_{i,2,G}, \ldots, v_{i,D,G}]$ of each individual i is generated according to

$$V_{i,G} = N(\mu_i, \sigma_i) \tag{1}$$

where $N(\mu_i, \sigma_i)$ is a Gaussian variable generator with mean $\mu_i = (X_{best,G} + X_{i,G})/2$ and standard deviation $\sigma_i = |X_{best,G} - X_{i,G}|$. $X_{best,G}$ is the best individual at generation G. At the early stage of the algorithm, $X_{best,G}$ and $X_{i,G}$ are far from each other, so that the mutation focuses on exploration of the entire space. Afterward, with the convergence of the individuals, the operation tends to do exploitation in a promising subspace of the problem.

After mutation, the binomial crossover is applied to generate a trial vector $U_{i,G} = [u_{i,1,G}, u_{i,2,G}, \ldots, u_{i,D,G}]$ using $X_{i,G}$ and $V_{i,G}$ as

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & \text{if } \text{rand}_j(0,1) < CR \text{ or } j = j_{\text{rand}} \\ x_{i,j,G}, & \text{otherwise} \end{cases} \tag{2}$$

where $\text{rand}_j(0, 1)$ is a uniformly random number in $(0, 1)$, $j_{\text{rand}} \in \{1, \ldots, D\}$ is a random dimension index, and CR is the crossover probability.

In the selection, the trial vector $U_{i,G}$ replaces $X_{i,G}$ if it has a better fitness value. Without loss of generality, we consider maximization problems here, so that solutions with larger objective values are preferred

$$X_{i,G+1} = \begin{cases} U_{i,G}, & \text{if } f(U_{i,G}) \geq f(X_{i,G}) \\ X_{i,G}, & \text{otherwise.} \end{cases} \tag{3}$$

The crossover probability CR plays an important role in the performance of GBDE [18]. A large CR accelerates the trial vector approaching the point $X_{best,G}$, but it may bring premature convergence. On the contrary, a small CR slows down the convergence speed. In this paper, we assign each individual with a CR_i and then use the adaptation strategy in [44] to adjust the parameter as

$$CR_i = N(\mu_{CR}, 0.1). \tag{4}$$

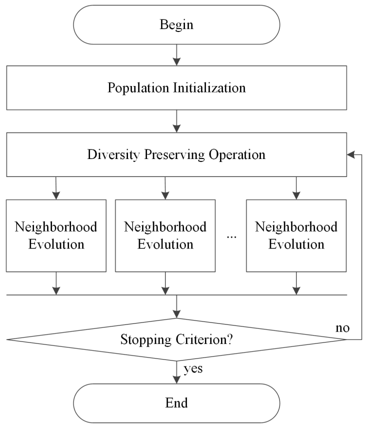


Fig. 2. Framework of BNDE.

It can be seen that CR_i is generated using Gaussian distribution with mean μ_{CR} and standard deviation 0.1. Moreover, the value should be truncated into [0, 1] when needed. The μ_{CR} in (4) is initialized as 0.5, and is updated in each generation according to

$$\mu_{CR,G+1} = (1 - q) \cdot \mu_{CR,G} + q \cdot \text{mean}(S_{CR}) \tag{5}$$

where S_{CR} is the set of CR_i that contributes to improved trial vectors in the current generation, mean$(\cdot)$ refers to the arithmetic mean, and $q \in [0, 1]$ is a constant that controls adaptation rate. In this way, promising settings of CR_i are propagated to the entire population as well as to the following generations. The crossover operation with good CR_i is more likely to generate trial vectors that would successfully improve the fitness value of individual i. According to the analysis in [44], the duration of a successful CR_i is $1/q$ generations, and it is suggested that $1/q \in [5, 20]$ can bring good performance for adapting the CR_i. Following the suggestion, in this paper, q is set to 0.1, namely, the μ_{CR} spans about 10 generations.

In this paper, we choose GBDE as our baseline algorithm owing to the following reasons. First, using a bare-bones EA, the proposed algorithm is free from fine-tuning the basic control parameters such as CR and F. Second, the Gaussian mutation in (1) can be considered as sampling individuals in a niche with center μ_i and standard deviation σ_i. By redefining μ_i and σ_i, the algorithm is possible to conduct exploitation in different niches so as to locate multiple optima.

IV. Proposed Niching Method

Based on the GBDE, the proposed BNDE algorithm further utilizes a neighborhood strategy and a diversity preserving operator (DPO) to realize MMOP. The framework of BNDE is sketched in Fig. 2. After random initialization, the algorithm enters an iteration that executes DPO and neighborhood evolution alternately. In order to avoid global convergence, the mean and standard deviation in the Gaussian mutation are defined locally within each neighborhood. DPO is applied to reinitialize converged and overlapped niches. In this way, the algorithm is able to reduce invalid or unnecessary search efforts and utilize the saved computational resources for global

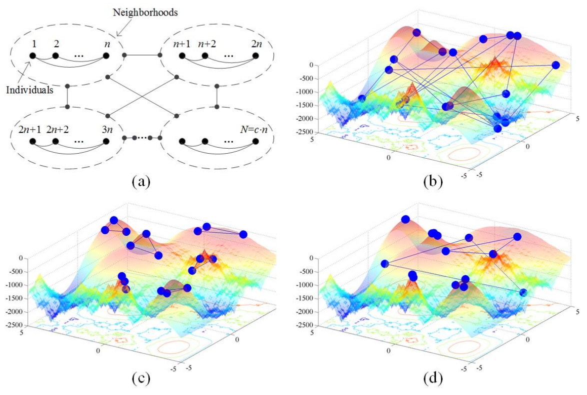


Fig. 3. Illustration of the neighborhoods and the different optimization states of BNDE ($n = 3$, for instance). (a) Index-based neighborhoods. (b) Exploration (initial) state. (c) Exploitation state. (d) Hybrid state.

exploration. Next, we are going to present the algorithm in detail.

A. Neighborhood Strategy

The neighborhood strategy of BNDE is shown in Fig. 3(a), where the links represent the intra- and inter-connections of neighborhoods. Each neighborhood is composed of n individuals and the total number of neighborhoods (subpopulations) in the population is $c = N/n$. As illustrated in Fig. 3(a), the neighborhoods are index-based and the individuals interact immediately with their neighbors. Then, the neighborhood, which is considered as a whole, influences each other in the DPO being described in the next section.

BNDE follows the basic procedure of classical DE algorithms, which performs mutation, crossover, and selection, iteratively, until the stopping criterion is met. The formulation of the three basic evolution operators is according to (1)–(3) in GBDE, where the only difference is the definition of μ_i and σ_i. In BNDE, the mean and standard deviation of the Gaussian mutation are defined as

$$\mu_i = X_{nbest,G} \tag{6}$$

and

$$\sigma_i = \begin{cases} |X_{nrand,G} - X_{i,G}|, & \text{if rand}(0,1) > PE \\ \chi \cdot (X_{\max} - X_{\min}), & \text{otherwise} \end{cases} \tag{7}$$

where $nbest$ and $nrand \neq i$ are the indices of the best individual and a random individual in the neighborhood, respectively; $X_{\max}$ and $X_{\min}$ refer to the upper and lower bounds of the problem space; PE is an extra probability that sets the standard deviation according to the variable range; and χ is a restriction coefficient. Both PE and χ are adjusted during the optimization process, which will be introduced later in this section.

From (6), it can be seen that, instead of using $\mu_i = (X_{nbest,G} + X_{i,G})/2$ according to the setting of μ_i in (1), BNDE adopts the best neighboring individual as the mutation center, for the following two reasons. First, in the Gaussian distribution, the sampling points are more likely to be located near the mean point. Using $X_{nbest,G}$ as the mean of Gaussian distribution, the search behavior of individuals is greedier, which increases the convergence speed of the neighborhood toward the optimum. Second, an essential method of BNDE is to locate different niches on different peaks in the landscape. As illustrated in Fig. 4(a), using $\mu_i = X_{nbest,G}$ is more natural for realizing this method, since the Gaussian distribution centered at $X_{nbest,G}$ matches well with the contour of a peak. In contrast, if $\mu_i = (X_{nbest,G} + X_{i,G})/2$ is used, as shown in Fig. 4(b), the niche may be located at the "valley" between the peaks of $X_{nbest,G}$ and $X_{i,G}$. The valley is an inferior area that will waste a great deal of search efforts to exploit/escape, which goes against our purpose of fast local contour matching. In MMOP, the average amount of computing resources available for locating each peak is more limited than that in traditional global optimization. In this sense, it is better to avoid using a Gaussian mutation center at the valley between peaks. Besides, we also made experimental comparisons between these two methods. The results verified our anticipation: using $\mu_i = X_{nbest,G}$ as the mutation center greatly enhances the exploitation ability and hence improves the solution accuracy.

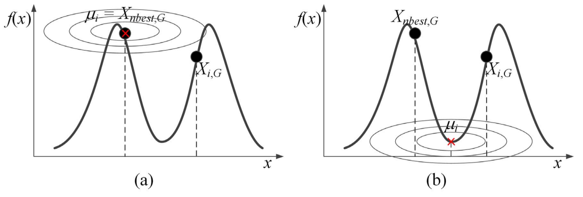


Fig. 4. Comparison of using different mutation centers. (a) $\mu_i = X_{nbest,G}$. (b) $\mu_i = (X_{nbest,G} + X_{i,G})/2$, when $X_{nbest,G}$ and $X_{i,G}$ are located on different peaks.

On the other hand, the standard deviation σ_i determines the shape of the Gaussian distribution and controls the relative force of exploitation and exploration for the individuals. According to the *three-sigma rule* [45], computed from the Gaussian cumulative distribution function, the probability of the sampled values locating within $[\mu_i - \sigma_i, \mu_i + \sigma_i]$, $[\mu_i - 2\sigma_i, \mu_i + 2\sigma_i]$, $[\mu_i - 3\sigma_i, \mu_i + 3\sigma_i]$, and the others are 68.27%, 95.45%, 99.73%, and 0.27%, respectively. In BNDE, σ_i is set to the difference between $X_{nrand,G}$ and $X_{i,G}$ in most cases. In this way, σ_i decreases with the convergence of neighborhood, which can be considered as adaptively adjusting the niche radius. Otherwise, within the probability PE, the standard deviation is set to a value proportional to the variable range for the purpose of robust exploration. The restriction coefficient χ decreases exponentially according to the ratio of the number of fitness evaluations (FEs) to the maximum allowed FEs ($MaxFEs$) in a deterministic way

$$\chi = \exp\left(-4 \cdot \left(\frac{FEs}{MaxFEs} + 0.4\right)\right). \tag{8}$$

The adaptation curve of χ is plotted in Fig. 5, which starts from 0.2 and decreases to 0.004 at the end of optimization. At the beginning, the robust exploration component allows individuals to explore the space with σ_i being 20% of the search range. It guarantees fully exploring the entire problem

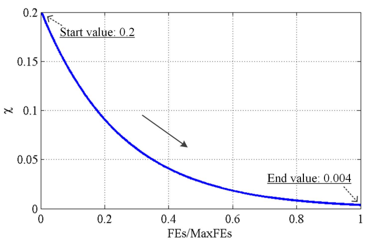


Fig. 5. Adaptation curve of χ.

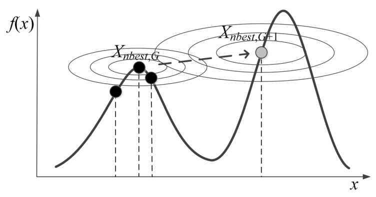


Fig. 6. Illustration of a niche moving to a better peak.

space for the sake of finding as more peaks as possible. Then, the value decreases quickly to less than 10% of the search range and eventually comes to 4‰ of the range. This mechanism protects the formulated population distribution from being changed too much by the robust exploration component and reduces the risk of replacement errors.

As for the probability parameter PE, it is adaptively adjusted in the same way as adapting the CR [44]. Namely, each individual is assigned with a PE_i generated by

$$PE_i = N(\mu_{PE}, 0.1) \tag{9}$$

$$\mu_{PE,G+1} = (1-q)\cdot\mu_{PE,G} + q\cdot\text{mean}(S_{PE}) \tag{10}$$

where μ_{PE} is initialized to 0.5; S_{PE} is the set of PE_i that contributes to improved individuals; and $q = 0.1$ is the adaptation rate. PE_i is truncated into [0, 1] when needed. In this way, useful PE_i values are propagated to the entire population as well as to the following generations.

Once an offspring is generated on another peak and is found to have better fitness than all the current individuals in the neighborhood, owing to the selection, $X_{nbest,G+1}$ is updated to the position on the better peak. As shown in Fig. 6, in such cases, the niche is immediately moved to a new peak. According to the selection defined in (3), the individuals can only move toward the better area (degradation is never allowed in DE). Therefore, once the niche moves to a new and better peak, it will never roll back.

Fig. 3(b)–(d) depicts three possible optimization states of the algorithm, namely, the exploration of the entire problem space, the exploitation of multiple peaks, and their hybrid. The exploration state appears at the initial stage of the optimization process, when individuals are randomly scattered in the problem space. At this stage, σ_i of all individuals in the population is large, so that the individual tends to explore the entire problem space. Afterward, with local evolution, the neighborhoods converge and conduct exploitation in different niches. The algorithm comes to the exploitation state. However, in most of the time, the population would be engaged in a hybrid state as shown in Fig. 3(d): a proportion of the neighborhoods performs local exploitation, whereas the others do global exploration. This is because: 1) the convergence speed of different neighborhoods differs and 2) the algorithm would merge and reinitialize some niches in the DPO.

B. Diversity Preservation

In MMOP, the multiple optima can be far from each other in the problem landscape. On one hand, diversity is crucial for a niching algorithm to locate such multiple optima. On the other hand, since the algorithm needs to approach a number of peaks simultaneously, the average computational resources available for approaching each peak are very limited. Because of these, we embed a diversity preserving operator (DPO) in BNDE, which not only maintains diversity, but also saves inefficient search efforts.

Considering each neighborhood, when all individuals have completely converged at the top of a peak, further exploitation on the peak is unnecessary and could be a waste of search efforts. At this moment, the DPO will record the current $X_{nbsest,G}$ in a specific memory (an external archive $\boldsymbol{A}$ maintained by the algorithm) and then reinitialize the neighborhood. Particularly, at the beginning of each generation G, the center C_k of each neighborhood is calculated ($k = 1, 2, \ldots, c$). Define a neighborhood radius $r_k = ||C_k, X_{nrand,G}||$ as the Euclidean distance between C_k and a random individual $X_{nrand,G}$ in the neighborhood. If $r_k = 0$, it is recognized that individuals in the neighborhood have converged to a single position. In such cases, DPO adds the best individual of the neighborhood to the external archive, reinitializes all individuals in the converged neighborhood, re-evaluates the individuals, and increases the FEs count by n. In the implementation, we use a small constant d_0 to replace the absolute zero distance. Namely, the neighborhood is considered as converged once $r_k \le d_0$. Without loss of generality, d_0 is set to $10^{(-16/||\mathbf{1}-\mathbf{0}||)}$, where 10^{-16} is the machine precision; $\mathbf{1}$ and $\mathbf{0}$ are unit and zero vectors in the D-dimensional space; and therefore $||\mathbf{1}-\mathbf{0}||$ denotes the unit distance in the D-dimensional Euclidean space. This is because using 10^{-16} directly is too critical when the problem space is large. This mechanism gradually relaxes the restart threshold with the increase of problem space.

In addition, if two neighborhoods in the population overlap, DPO will merge the neighborhoods so as to save computational resources for further global exploration. The process is as follows. First, $d_{k_1,k_2} = ||C_{k_1}, C_{k_2}||$ is defined as the Euclidean distance between neighborhoods k_1 and k_2. If $d_{k_1,k_2} < \xi$, where ξ is an overlapping threshold, k_1 and k_2 can be merged. Suppose $(nbest_1, nworst_1)$ and $(nbest_2, nworst_2)$ are the indices of the best and worst individuals in the two neighborhoods, respectively. We will merge the neighborhood with worse $nbest$ fitness into the better neighborhood. Assuming now we have $f(X_{nbest_1,G}) \ge (X_{nbest_2,G})$, the merging process is illustrated in Fig. 7. If $f(X_{nbest_2,G}) \ge (X_{nworst_1,G})$, $X_{nworst_1,G}$ is replaced by $X_{nbest_2,G}$ and the structure of neighborhood k_1 is updated. Otherwise, neighborhood k_1 dominates k_2. In this case, no replacement is conducted. Finally, after merging the two neighborhoods, all individuals

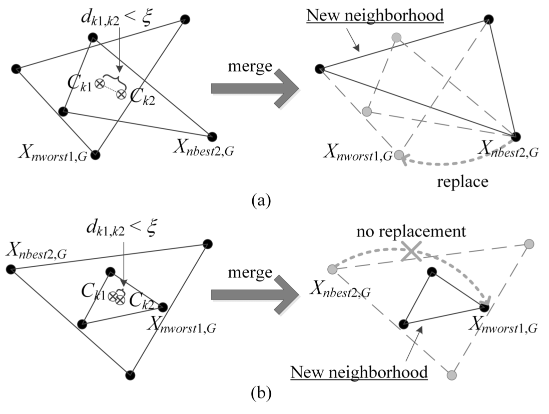


Fig. 7. Illustration of merging overlapped neighborhoods ($n = 3$, for instance). (a) Case one: $f(X_{nbest_2,G}) \geq f(X_{nworst_1,G})$. (b) Case two: $f(X_{nbest_2,G}) < f(X_{nworst_1,G})$.

in k_2 are randomly reinitialized. Every time an individual is reinitialized, we evaluate the fitness and increase the *FEs* count immediately.

Generally speaking, the proposed DPO is composed of two phases, one for reinitializing converged niches and the other for merging overlapped neighborhoods. In this way, some invalid or inefficient search efforts are saved and are thus utilized for the exploration of other possible optima. The pseudocode of DPO is presented in **Algorithm 1**. The computational cost introduced by such niching operation is $O(c^2)$.

C. Overall Procedure of BNDE

To summarize, BNDE employs the bare-bones DE as the baseline optimization algorithm. Gaussian mutation is specifically designed, by which the algorithm cannot only focus its search force on exploiting the most promising subspaces but also has probability to jump out of inferior local optima. The design/introducing of index-based neighborhoods avoids extra computational overhead and excludes the niching parameters such as the niche radius (which is very sensitive to the landscapes). DPO is embedded in the algorithm for two goals: increasing diversity and improving search efficiency.

The overall procedure of BNDE is shown in **Algorithm 2**. It can be seen that the proposed algorithm follows the same framework of the GBDE algorithm. The two modifications are: 1) at each generation, DPO is performed before the evolution of individuals and 2) the mutation and crossover are performed locally within the neighborhoods. At the end of the algorithm, the external archive $\boldsymbol{A}$ as well as the neighborhood best individuals in the final population $\boldsymbol{P}$ are outputted. Then, a standard procedure for optima identification (the Algorithm 1 in [46]) is performed on the set of output solutions to obtain the final solution set.

Algorithm 1 Diversity-Preserving Operation (DPO)

```
1: $S = \varnothing$;
2: for $i = 1$ to $c$ do
3:   if $i \in S$ then
4:     Continue; (i.e., go to line 2)
5:   end if
6:   if $(r_k = ||C_i, X_{nrand,G}||) \leq d_0$ then
7:     Record the $X_{nbest,G}$ of neighborhood $i$ in $\boldsymbol{A}$;
8:     Reinitialize all individuals in neighborhood $i$;
9:     Evaluate the reinitialized individuals;
10:    $S = S \cup \{i\}$;
11:  end if
12:  for $j = i + 1$ to $c$ do
13:    if $j \in S$ or $(d_{k_1,k_2} = ||C_i, C_j||) > \xi$ then
14:      Continue; (i.e., go to line 12)
15:    end if
16:    Determine $nbest_i, nworst_i, nbest_j, nworst_j$;
17:    if $f(X_{nbest_i,G}) \geq f(X_{nbest_j,G})$ then
18:      if $f(X_{nbest_j,G}) \geq f(X_{nworst_i,G})$ then
19:        $X_{nworst_i,G} = X_{nbest_j,G}$;
20:      end if
21:      Reinitialize all individuals in neighborhood $j$;
22:      Evaluate the reinitialized individuals;
23:      $S = S \cup \{j\}$;
24:    else
25:      if $f(X_{nbest_i,G}) \geq (X_{nworst_j,G})$ then
26:        $X_{nworst_j,G} = X_{nbest_i,G}$;
27:      end if
28:      Reinitialize all individuals in neighborhood $i$;
29:      Evaluate the reinitialized individuals;
30:      $S = S \cup \{i\}$;
31:      Break; (i.e., go to line 2)
32:    end if
33:  end for
34: end for
```

Algorithm 2 Bare-Bones Niching DE (BNDE)

```
1: $G = 0$;
2: Randomly initialize $N$ individuals;
3: while (stopping criterion is not satisfied) do
4:   Perform DPO according to Algorithm 1;
5:   for $i = 1$ to $N$ do
6:     Compute $\mu_i$ and $\sigma_i$ according to (6) and (7);
7:     Generate mutate vector $V_{i,G}$ according to (1);
8:     Generate trial vector $U_{i,G}$ according to (2);
9:     Evaluate $U_{i,G}$;
10:    Perform selection according to (3);
11:  end for
12:  Update $CR$ and $PE$ according to (4) and (9);
13:  $G = G + 1$;
14: end while
15: Output the final solution set by a standard procedure.
```

V. Numerical Experiments

A. Experimental Setup

In this section, the effectiveness of the proposed algorithm is verified through a state-of-the-art benchmark test suite adopted in the 2013, 2015, and 2016 IEEE competitions on niching methods [46]. The test suite is composed of 20 problems that not only cover a wide variety of multimodal landscapes but also possess good discrimination for different niching

TABLE I
CHARACTERISTICS OF THE 20 BENCHMARK FUNCTIONS

Problem ID	Name	Characteristics	No. global optima
F_1 (1D)	Five-Uneven-Peak Trap	Simple, deceptive	2
F_2 (1D)	Equal Maxima	Simple	5
F_3 (1D)	Uneven Decreasing Maxima	Simple	1
F_4 (2D)	Himmelblau	Simple, non-scalable, non-symmetric	4
F_5 (2D)	Six-Hump Camel Back	Simple, not-scalable, non-symmetric	2
F_6 (2D, 3D)	Shubert	Scalable, unevenly distributed grouped optima, numerous local optima	18, 81
F_7 (2D, 3D)	Vincent	Scalable, vastly different distance between optima	36, 216
F_8 (2D)	Modified Rastrigin	Scalable, symmetric	12
F_9 (2D)	Composition Function 1	Scalable, separable near the global optima, non-symmetric, numerous local optima	6
F_{10} (2D)	Composition Function 2	Scalable, separable near the global optima, non-symmetric, numerous local optima	8
F_{11} (2D, 3D, 5D, 10D)	Composition Function 3	Scalable, non-separable, non-symmetric, a huge number of local optima	6, 6, 6, 6
F_{12} (3D, 5D, 10D, 20D)	Composition Function 4	Scalable, non-separable, non-symmetric, a huge number of local optima	8, 8, 8, 8

methods. As described in Table I, the first five instances are low-dimensional simple multimodal functions, the next five instances are scalable functions with a large amount of global optima, and the last ten are composition functions that mix different functions together. The main characteristics of these instances are also presented in Table I.

The proposed BNDE algorithm is compared with some representative and state-of-the-art niching algorithms listed as follows:

1) CDE [14]: crowding DE;
2) SDE [15]: speciation-based DE;
3) NCDE [41]: neighborhood-based CDE;
4) NSDE [41]: neighborhood-based SDE;
5) DE/nrand/1 [47]: DE with nearest-neighbor mutation;
6) dADE/nrand/1 [48]: The DE/nrand/1 with dynamic archive and adaptive parameters;
7) DE/inrand/1R [40]: Differential evolution with ring neighborhood;
8) PNPCDE [42]: Proximity-based crowding DE with parent-centric normalized mutation;
9) LoISDE [43]: locally informative speciation DE;
10) FER-PSO [49]: PSO with fitness Euclidean-distance ratio;
11) R3PSO-lhc [39]: Local PSO with ring-3 neighborhood and local hill climbers;
12) LIPS [16]: locally informed particle swarm.

All algorithms are allowed to conduct a certain number of function evaluations (*MaxFEs*) according to [46]. To be specific, *MaxFEs* is set to 50 000 for F_1 to F_5 (1D or 2D), 200 000 for F_6 to F_{11} (2D), and 400 000 for F_6 to F_{12} (3D or more). We run the algorithms 50 times independently on the same platform. The parameter settings of the compared algorithms, such as the F and *CR*, niche radius, neighborhood size, and population size, follow the suggestions of their original papers listed above. For BNDE, basic parameters like F and *CR* are avoided using the bare-bones baseline algorithm, but it also introduces a niching parameter, namely, the overlapping threshold ξ in DPO. A relatively small number for the threshold is suggested. In the experiments, it is set as $\xi = 0.01$ for all instances. Besides, the population and neighborhood sizes are simply set according to the problem dimensions: $(N, n) = (150, 3)$, $(600, 3)$, and $(600, 18)$ for $D < 3$, $3 \leq D < 20$, and $D \geq 20$, respectively.

TABLE II
TIME EFFICIENCY OF NICHING METHODS

Category	Time Complexity per Generation[1]	Algorithm	Execution Time for CEC 2013 (in minutes)[2]
I	$O(N)$	R3PSO-lhc	87.23
II	$O(N \cdot n)$	DE/inrand/1R	96.09
	$O(c^2)$	BNDE	97.08
	$O(N^2)$	DE/nrand/1	98.29
		CDE	124.38
		FER-PSO	152.84
		PNPCDE	358.48
III	$O(N^2 + N \cdot \log N)$	SDE	277.87
		LoISDE	278.88
	$O(N^2 + N \cdot P)$	dADE/nrand/1	550.21
IV	$O(N^2 \cdot n)$	LIPS	162.41
		NSDE	667.65
		NCDE	1244.93

[1] N denotes the population size; c denotes the number of subpopulations; n denotes the neighborhood size; P denotes the archive size; D denotes the number of dimensions of the problem.
[2] The summation of execution time for running 50 times of each algorithm on the 20 problems.

B. Note on the Time Efficiency

Before presenting the experimental results, the time efficiency of the above algorithms is compared in this section. Listed in Table II, we roughly categorize the algorithms into four groups according to their theoretical time complexity.

1) First, R3PSO-lhc [39] is extremely fast with $O(N)$ per generation, which is identical with the complexity of classical PSO algorithm. Our experimental results also show that the algorithm has the shortest execution time among all the compared algorithms on the test suite.
2) Second, BNDE, DE/inrand/1R [40], DE/nrand/1 [47], CDE [14], FER-PSO [49], and PNPCDE [42] are considered as fast niching algorithms, because their computational complexity is less than or equal to $O(N^2)$.
3) Third, the speciation-based SDE [15] and LoISDE [43] need to sort the population and calculate the distance between individuals, which are thus $O(N^2 + N \cdot \log N)$ complex. The dADE/nrand/1 algorithm [48] computes the distance between individual pairs in the population as well as the distance between the updated solution and each potential solution restored in the dynamic archive. The computational complexity is $O(N^2 + N \cdot P)$. These three algorithms are considered to have medium computational complexity.

TABLE III
COMPARISONS OF THE MEAN AND MEDIAN PR OBTAINED BY THE 13 ALGORITHMS (INTEGERS PLACED IN THE PARENTHESES INDICATE THE CORRESPONDING RANKS)

Accuracy level ε		BNDE	CDE	SDE	NCDE	NSDE	DE/ nrand/1	dADE/ nrand/1	DE/ inrand/1R	PNPCDE	LoISDE	FER-PSO	R3PSO-lhc	LIPS
1.00E-01	Mean	**0.943 (1)**	0.621 (10)	0.592 (12)	0.746 (4)	0.691 (6)	0.618 (11)	0.827 (2)	0.635 (9)	0.793 (3)	0.713 (5)	0.641 (8)	0.524 (13)	0.678 (7)
	Median	**1.000 (1)**	0.944 (4)	0.648 (11)	0.947 (3)	0.712 (9)	0.630 (13)	0.942 (5)	0.652 (10)	0.994 (2)	0.811 (7)	0.918 (6)	0.635 (12)	0.790 (8)
1.00E-02	Mean	**0.784 (1)**	0.517 (12)	0.587 (8)	0.581 (9)	0.618 (6)	0.611 (7)	0.735 (2)	0.627 (5)	0.557 (10)	0.641 (4)	0.512 (13)	0.517 (11)	0.672 (3)
	Median	**0.890 (1)**	0.553 (12)	0.645 (9)	0.646 (8)	0.668 (5)	0.625 (11)	0.702 (3)	0.652 (7)	0.543 (13)	0.668 (5)	0.698 (4)	0.632 (10)	0.782 (2)
1.00E-03	Mean	**0.771 (1)**	0.476 (12)	0.584 (8)	0.549 (9)	0.611 (5)	0.602 (6)	0.725 (2)	0.618 (4)	0.496 (11)	0.594 (7)	0.414 (13)	0.513 (10)	0.666 (3)
	Median	**0.866 (1)**	0.537 (11)	0.645 (6)	0.591 (9)	0.668 (4)	0.623 (8)	0.705 (3)	0.649 (5)	0.386 (12)	0.583 (10)	0.311 (13)	0.630 (7)	0.779 (2)
1.00E-04	Mean	**0.761 (1)**	0.457 (11)	0.579 (7)	0.536 (9)	0.606 (5)	0.595 (6)	0.709 (2)	0.612 (4)	0.439 (12)	0.552 (8)	0.305 (13)	0.511 (10)	0.662 (3)
	Median	**0.842 (1)**	0.421 (11)	0.643 (5)	0.561 (9)	0.667 (4)	0.623 (8)	0.696 (3)	0.642 (6)	0.225 (12)	0.511 (10)	0.072 (13)	0.625 (7)	0.774 (2)
1.00E-05	Mean	**0.747 (1)**	0.442 (11)	0.576 (7)	0.512 (8)	0.599 (5)	0.590 (6)	0.695 (2)	0.605 (4)	0.412 (12)	0.498 (10)	0.231 (13)	0.507 (9)	0.658 (3)
	Median	**0.811 (1)**	0.341 (11)	0.643 (5)	0.498 (9)	0.667 (4)	0.622 (7)	0.692 (3)	0.638 (6)	0.036 (12)	0.417 (10)	0.010 (13)	0.618 (8)	0.763 (2)
Total	Mean	**0.793 (1)**	0.503 (12)	0.584 (9)	0.585 (8)	0.625 (4)	0.603 (6)	0.738 (2)	0.619 (5)	0.540 (10)	0.599 (7)	0.421 (13)	0.514 (11)	0.667 (3)
	Median	**0.882 (1)**	0.559 (11)	0.645 (7)	0.649 (5)	0.676 (4)	0.625 (9)	0.747 (3)	0.647 (6)	0.437 (12)	0.598 (10)	0.402 (13)	0.628 (8)	0.778 (2)

4) Forth, considering LIPS [16], NSDE [41], and NCDE [41], the complexity of finding the n nearest neighbors for all individuals is $O(N^2 \cdot n)$. Note that the actual execution time of some algorithms in the same group differs a lot. For example, although theoretically possessing the same complexity, NCDE requires much longer execution time than the LIPS. This is mainly because of the different parameter settings of the two algorithms. For the sake of good performance, NCDE requires much larger population and neighborhood sizes than LIPS, as can be observed from the parameter settings in the original papers of these two algorithms [16], [41].

From the above analysis, it can be seen that the proposed BNDE algorithm possesses a very high time efficiency in terms of both theoretical computational complexity and actual execution time.

C. Performance Measures

We use two performance measures that are commonly used in the literature [46], namely, the peak ratio (PR) and the successful rate (SR), to evaluate the performance of the algorithms. Given an accuracy level ε, PR is calculated as

$$\mathrm{PR} = \frac{\sum_{i=1}^{\mathrm{NR}} \mathrm{NPF}_i}{\mathrm{NKP} \cdot \mathrm{NR}}. \tag{11}$$

Here NPF_i is the number of optima found by the algorithm in its ith run when considering a specific accuracy level. A standard procedure for determining such optima is provided in [46]. Note that the procedure is only used at the end of the algorithm to identify the solutions. Besides, NKP stands for the number of known global peaks for the problem and NR is the number of runs. It can be seen that PR measures the ability of an algorithm to find multiple global optima for the problems, while a thresholding parameter ε is used to determine the level of accuracy. Five accuracy levels are used in the experiments, i.e., $\varepsilon \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. In general, the higher the accuracy level is, the lower PR the algorithm can achieve.

Besides, SR provides a strict way to examine the algorithms' capability to achieve the complete coverage of global optima, which is defined as

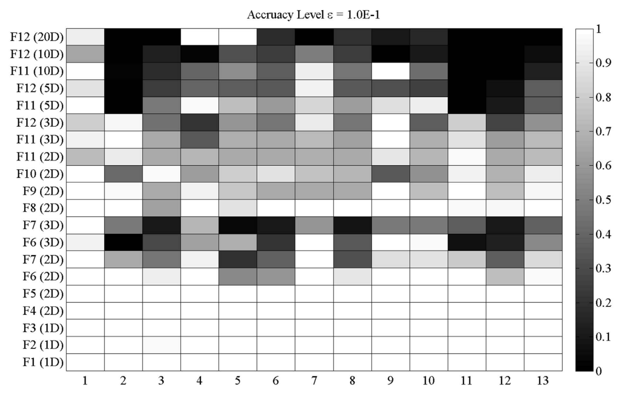
$$\mathrm{SR} = \frac{\mathrm{NSR}}{\mathrm{NR}} \tag{12}$$

where NSR refers to the number of successful runs that the algorithm finds all global optima for the problem. SR is more strict than PR since $\mathrm{NSR} \leq ((\sum_{i=1}^{\mathrm{NR}} \mathrm{NPF}_i)/(\mathrm{NKP}))$. Moreover, SR is also highly related to the requirement of accuracy. Specifically, the measure is important when applying MMOP to interpret the distribution of models (the complete set of global optima is needed).

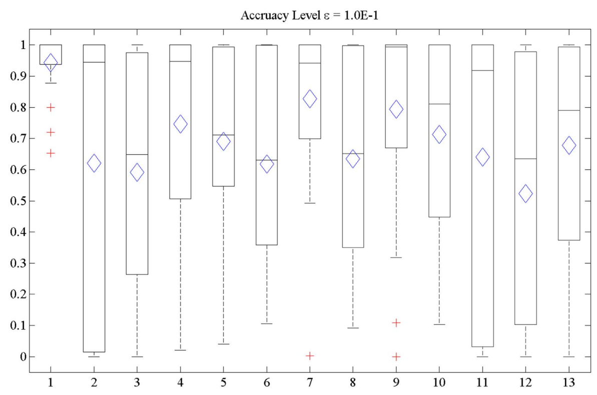
D. Experimental Results and Comparisons

Table III reports the mean and median results obtained by the thirteen algorithms, with an increasing accuracy level. The ranks of performance are placed in the parentheses. Moreover, the total results regarding to the average of values for all accuracy levels are summarized at the bottom lines of Table III. From a very general view, it can be seen that the proposed BNDE algorithm ranks the first among all compared algorithms, followed by dADE/nrand/1, LIPS, and NSDE. Besides, PNPCDE and LoISDE possess good performance at low accuracy levels. Here, we use pseudo-color plots and box plots to visualize the results. Meanwhile, the Wilcoxon rank-sum tests [50]–[52] are conducted to show the significance of differences between BNDE and the other algorithms.

Considering $\varepsilon = 10^{-1}$, as shown in Fig. 8(a), the proposed BNDE algorithm can achieve PR $=$ 100% for 13 out of the 20 instances and PR $\geq$ 90% for 16 instances. In the box plot in Fig. 8(b), the mean, median, upper and lower quartiles and extremes, whiskers, and outliers are depicted, which illustrate that the stability of BNDE is the best among all the algorithms. Particularly, for the simple functions F_1 to F_5, all the algorithms perform very well that they obtain PR equal or very close to 100%. However, for the problems with much complex landscapes and more global optima, the performance of many compared algorithms decreases sharply, whereas only a few algorithms maintain the capability of locating diverse solutions. For example, the problems F_6(3D) and F_7(3D) have the most global optima among all problems, i.e., NKP $=$ 81 and 216, respectively. For F_6(3D), only four algorithms, BNDE, dADE/nrand/1, PNPCDE, and LoISDE,

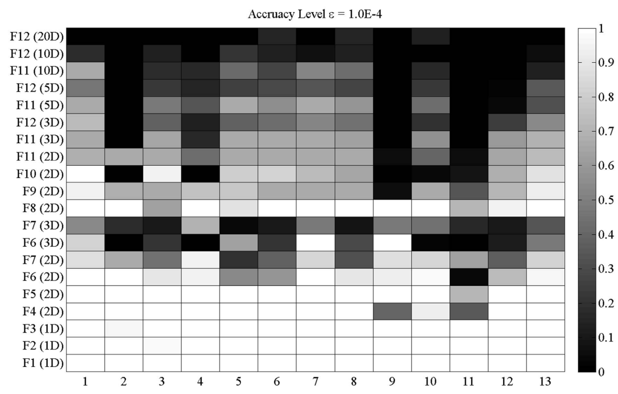


(a)

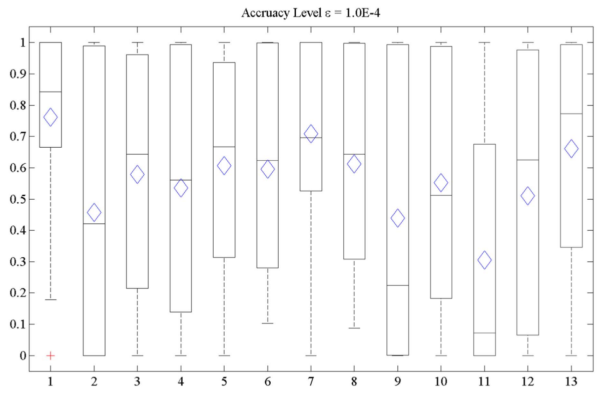


(b)

Fig. 8. Comparisons of the PR values at accuracy level $\varepsilon = 10^{-1}$ (numbers in the horizontal axis indicate the results of different algorithms: 1-BNDE, 2-CDE, 3-SDE, 4-NCDE, 5-NSDE, 6-DE/nrand/1, 7-dADE/nrand/1, 8-DE/inrand/1R, 9-PNPCDE, 10-LoISDE, 11-FER-PSO, 12-R3PSO-lhc, and 13-LIPS). (a) Pseudo-color plot. (b) Box plot.



(a)



(b)

Fig. 9. Comparisons of the PR values at accuracy level $\varepsilon = 10^{-4}$ (numbers in the horizontal axis indicate the results of different algorithms: 1-BNDE, 2-CDE, 3-SDE, 4-NCDE, 5-NSDE, 6-DE/nrand/1, 7-dADE/nrand/1, 8-DE/inrand/1R, 9-PNPCDE, 10-LoISDE, 11-FER-PSO, 12-R3PSO-lhc, and 13-LIPS). (a) Pseudo-color plot. (b) Box plot.

TABLE IV

WILCOXON RANK-SUM TEST RESULTS (WTRs) BETWEEN BNDE AND THE OTHER ALGORITHMS AT ACCURACY LEVEL $\varepsilon = 10^{-1}$

WTRs (sig-better / non-sig / sig-worse)	vs CDE	vs SDE	vs NCDE
	10 / 6 / 4	16 / 4 / 0	11 / 8 / 1
	vs NSDE	vs DE/nrand/1	vs dADE/nrand/1
	14 / 5 / 1	14 / 6 / 0	9 / 8 / 3
	vs DE/inrand/1R	vs PNPCDE	vs LoISDE
	15 / 5 / 0	7 / 9 / 4	11 / 8 / 1
	vs FER-PSO	vs R3PSO-lhc	vs LIPS
	10 / 9 / 1	15 / 5 / 0	14 / 6 / 0

TABLE V

WILCOXON RANK-SUM TEST RESULTS (WTRs) BETWEEN BNDE AND THE OTHER ALGORITHMS AT ACCURACY LEVEL $\varepsilon = 10^{-4}$

WTRs (sig-better / non-sig / sig-worse)	vs CDE	vs SDE	vs NCDE
	12 / 7 / 1	14 / 6 / 0	11 / 7 / 2
	vs NSDE	vs DE/nrand/1	vs dADE/nrand/1
	10 / 8 / 2	12 / 7 / 1	8 / 11 / 1
	vs DE/inrand/1R	vs PNPCDE	vs LoISDE
	14 / 5 / 1	12 / 7 / 1	14 / 5 / 1
	vs FER-PSO	vs R3PSO-lhc	vs LIPS
	16 / 4 / 0	14 / 6 / 0	10 / 9 / 1

obtain PR near 100%. For F_7(3D), only BNDE achieves PR of 100%, whereas the values obtained by all the other algorithms are lower than 80%. These results verify the strong global exploration ability of the proposed algorithm. When considering the composition problems that have extremely complex landscapes, the advantage of BNDE compared with other algorithms becomes more prominent. For example, the proposed algorithm achieves PR = 100% for four composition instances and PR > 80% for another four instances, whereas most of the other algorithms obtain unsatisfactory results on these problems. It has been shown that BNDE exhibits promising search performance in such difficult problem spaces.

We also conduct significance tests between BNDE and each other algorithm to further compare their performance. The one-tail Wilcoxon rank-sum tests with 98 degrees of freedom at significance level $\alpha = 0.05$ is applied. Table IV reports the number of problems that BNDE obtains significantly better or worse results than the compared algorithms, as well as the number of problems that the results are not significant. It can be seen from the table that, for most ($\geq$16/20) problems, BNDE can arrive at significantly better or competitive results when compared with the others. Only for a very small fraction of ($\leq$4/20) problems, some other algorithm significantly outperforms BNDE.

TABLE VI
COMPARISONS OF THE MEAN SR ON THE TEST SUITE WITH DIFFERENT ACCURACY LEVELS
(INTEGERS PLACED IN THE PARENTHESES INDICATE THE CORRESPONDING RANKS)

Accuracy level ε	BNDE	CDE	SDE	NCDE	NSDE	DE/ nrand/1	dADE/ nrand/1	DE/ inrand/1R	PNPCDE	LoISDE	FER-PSO	R3PSO-lhc	LIPS
1.00E-01	**0.762 (1)**	0.513 (5)	0.303 (12)	0.499 (6)	0.347 (9)	0.314 (10)	0.611 (3)	0.314 (10)	0.654 (2)	0.406 (7)	0.529 (4)	0.278 (13)	0.386 (8)
1.00E-02	**0.442 (1)**	0.382 (4)	0.299 (10)	0.367 (6)	0.270 (13)	0.313 (8)	0.409 (2)	0.313 (8)	0.399 (3)	0.351 (7)	0.297 (11)	0.278 (12)	0.373 (5)
1.00E-03	**0.439 (1)**	0.351 (6)	0.296 (10)	0.351 (5)	0.270 (12)	0.308 (9)	0.400 (2)	0.312 (8)	0.378 (3)	0.347 (7)	0.254 (13)	0.278 (11)	0.367 (4)
1.00E-04	**0.437 (1)**	0.342 (4)	0.288 (10)	0.324 (5)	0.270 (12)	0.307 (9)	0.400 (2)	0.311 (8)	0.318 (7)	0.324 (5)	0.194 (13)	0.277 (11)	0.361 (3)
1.00E-05	**0.432 (1)**	0.313 (4)	0.285 (10)	0.297 (8)	0.269 (12)	0.306 (6)	0.400 (2)	0.308 (5)	0.293 (9)	0.298 (7)	0.172 (13)	0.276 (11)	0.357 (3)
Total	**0.495 (1)**	0.380 (4)	0.294 (10)	0.368 (6)	0.285 (12)	0.310 (9)	0.444 (2)	0.312 (8)	0.408 (3)	0.345 (7)	0.289 (11)	0.277 (13)	0.369 (5)

The results of different algorithms at accuracy level $\varepsilon = 10^{-4}$ are presented in Fig. 9. The increase of accuracy level poses a more strict requirement of the exploitation ability of the solvers. At this accuracy level, for example, the rank of PNPCDE decreases, whereas the rank of LIPS increases. It means that PNPCDE has better exploration than LIPS, but LIPS wins in exploitation. It can be seen from the results that BNDE still performs the best in most cases, which possesses both good exploration and exploitation abilities. Moreover, the significance test results at this accuracy level are reported in Table V, which show that BNDE significantly outperforms the others on more problems.

Besides, let us take a closer look at the comparison between BNDE and DE/inrand/1R. Both algorithms are developed from DE with index-based neighborhood. But our BNDE algorithm has two major distinctions compared with DE/inrand/1R: using bare-bone Gaussian mutation and embedding DPO. The Gaussian mutation uses local mean and standard deviation according to (6) and (7), which facilitates exploiting promising niches. In addition, by embedding DPO, the global exploration and search efficiency are further enhanced. In this way, BNDE significantly outperforms DE/inrand/1R on most problem instances.

Table VI reports the mean SR values obtained by the 13 algorithms at different accuracy levels ε. As introduced above, SR measures the ability of an algorithm to detect all global optima, which is more strict than PR. In Table VI, BNDE achieves the highest SR values at all accuracy levels, followed by dADE/nrand/1 and PNPCDE. These algorithms are more reliable for application scenarios when all global optima of the problem should be detected (e.g., to provide geometric interpretation of the IBM-2 model [8]). Note that, the dADE/nrand/1 algorithm uses the parameter ε as prior knowledge in the optimization. Thus, for each accuracy level, the algorithm should run once. On the contrary, the proposed BNDE utilizes neither this information, nor the others such as the niche radius or the actual number of optima, and it is a more general algorithm.

E. Investigation of Parameters

In this section, we investigate the sensitivity of BNDE to three parameters: the population size N, the neighborhood size n, and the overlapping threshold ξ. Five representative problems are tested in the experiments: F_4(2D) is low dimensional and simple, F_6(3D) and F_7(3D) are problems containing a huge number of optima, and F_{11}(5D) and F_{12}(20D) are higher dimensional complex composition problems. Tables VII to IX report the average number of optima found by executing 50 runs of BNDE with different parameter settings.

1) Effect of the Population Size N: In the literature of MMOP, many algorithms fine-tune the population size for different problems [16], [17], [39], [41]–[43]. Very commonly, these algorithms use a large N if the problem has a large number of global optima, or use a much smaller N for problems containing fewer peaks. However, in practice, it is hard to know the number of global optima in advance. Using the "trial-and-error" method to select a proper N is time-consuming. In the above comparison, BNDE does not fine-tune this parameter, but it simply uses $N = 150$ for 1D and 2D problems and adopts $N = 600$ for higher dimensional problems. Therefore, the above results show the general performance of BNDE without overfitting to different problem instances. In order to make an investigation on the effect of population size in BNDE, we test six different values: $N \in \{60, 150, 300, 450, 600, 900\}$. The results are reported in Table VII. For problems F_6(3D) and F_7(3D) that have 81 and 216 peaks, it is desired to use a large population size. For the other problems with fewer than 10 peaks, using a small population size helps to improve the performance. Nevertheless, BNDE is not very sensitive to this parameter such that setting N in the range of $[150, 900]$ can bring promising results.

2) Effect of the Neighborhood Size n: In BNDE, different neighborhoods search different peaks in the problem landscape (otherwise they will be merged by the DPO). Therefore, the number of niches being exploited simultaneously by the population is $c = N/n$. We suggest to use a very small n so as to find as more niches as possible. Here, a set of neighborhood sizes $n \in \{3, 6, 9, 12, 15, 18\}$ is further investigated. As reported in Table VIII, for most of the test problems, the number of peaks found by BNDE decreases with the increase in n, which complies with our anticipation. But for F_{12}(20D), the peak identification capability of BNDE improves drastically by increasing n from 3 to 18. This may be because the small neighborhood cannot meet the requirement of exploitation in such a high-dimensional space and thus can hardly reach the accuracy level. Therefore, we suggest enlarging the neighborhood when dealing with high-dimensional problems.

3) Effect of the Overlapping Threshold ξ: The DPO of BNDE has an overlapping threshold parameter, which is set as $\xi = 0.01$ in the above experiments. Here, we fur-

TABLE VII

AVERAGE NUMBER OF PEAKS FOUND USING DIFFERENT POPULATION SIZES N

Accuracy level ε	Problem ID	NKP	Population Size N 60	150	300	450	600	900
1.0E-01	F_4(2D)	4	4.00	4.00	4.00	4.00	4.00	4.00
	F_6(3D)	81	42.30	60.30	69.74	73.46	76.24	78.54
	F_7(3D)	216	160.78	190.46	215.24	216.00	216.00	216.00
	F_{11}(5D)	6	5.86	5.80	5.84	6.00	6.00	6.00
	F_{12}(20D)	8	1.42	1.74	3.74	8.00	7.48	1.10
1.0E-04	F_4(2D)	4	4.00	4.00	4.00	4.00	4.00	3.88
	F_6(3D)	81	26.30	45.46	57.56	63.48	66.40	69.28
	F_7(3D)	216	78.86	92.98	104.50	112.78	117.50	124.10
	F_{11}(5D)	6	4.00	4.00	4.00	4.00	4.00	4.00
	F_{12}(20D)	8	1.42	1.58	1.12	0.06	0.02	0.00

TABLE VIII

AVERAGE NUMBER OF PEAKS FOUND USING DIFFERENT NEIGHBORHOOD SIZES n

Accuracy level ε	Problem ID	NKP	Neighborhood Size n 3	6	9	12	15	18
1.0E-01	F_4(2D)	4	4.00	4.00	4.00	4.00	4.00	3.98
	F_6(3D)	81	76.24	62.58	49.62	42.18	37.30	32.28
	F_7(3D)	216	216.00	209.58	193.30	182.16	176.46	171.12
	F_{11}(5D)	6	6.00	5.80	5.32	4.72	5.00	4.56
	F_{12}(20D)	8	0.82	2.32	3.42	5.38	6.72	7.48
1.0E-04	F_4(2D)	4	4.00	4.00	4.00	3.98	3.98	3.84
	F_6(3D)	81	66.40	60.10	46.60	38.14	32.02	26.88
	F_7(3D)	216	117.50	86.66	74.76	65.16	61.06	57.24
	F_{11}(5D)	6	4.00	4.00	4.00	4.00	4.00	3.94
	F_{12}(20D)	8	0.00	0.00	0.00	0.00	0.00	0.02

ther investigate the algorithm by varying ξ in $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$. The results are reported in Table IX. It can be observed from Table IX that, generally, the performance of BNDE is not sensitive to the overlapping threshold. We suggest using $\xi \in [0.01, 0.1]$, which brings the best results.

F. Investigation of DPO

To analyze the effectiveness of the DPO, we compare BNDE with its simplified version, termed BNDE-nd, by removing the use of DPO. The comparison results are reported in Table X. It can be observed that, without DPO, the average number of peaks found by BNDE-nd is smaller than that of BNDE. For example, the original BNDE finds all the 216 global optima of F_7(3D), whereas BNDE-nd obtains about 175. The results indicate that the DPO keeps the population diversity during the search process, which greatly enhances the global exploration ability of BNDE. We can make two conclusions from this investigation: 1) using the proposed neighborhood evolution strategy only, BNDE-nd is able to approach a number of solutions simultaneously and is applicable to MMOP and 2) by embedding the DPO, the performance of BNDE in locating all global optima can be further improved.

VI. LEARNING MULTIMODAL PARAMETERS FOR NEURAL NETWORK ENSEMBLE

The above experiments validate the state-of-the-art performance of the proposed algorithm for MMOP. In this section, we make an attempt to apply BNDE for training the neural network ensembles (NNEs). We first discuss the method and

TABLE IX

AVERAGE NUMBER OF PEAKS FOUND USING DIFFERENT OVERLAPPING THRESHOLD VALUES ξ

Accuracy level ε	Problem ID	NKP	Overlapping Threshold ξ 0.001	0.005	0.01	0.05	0.1	0.5
1.0E-01	F_4(2D)	4	4.00	4.00	4.00	4.00	4.00	4.00
	F_6(3D)	81	74.30	74.88	76.24	76.80	77.10	77.02
	F_7(3D)	216	216.00	216.00	216.00	216.00	216.00	216.00
	F_{11}(5D)	6	6.00	6.00	6.00	6.00	5.98	5.70
	F_{12}(20D)	8	7.52	7.58	7.48	1.04	0.10	0.02
1.0E-04	F_4(2D)	4	4.00	4.00	4.00	4.00	4.00	3.56
	F_6(3D)	81	60.86	63.92	66.40	68.76	70.38	71.46
	F_7(3D)	216	110.16	113.90	117.50	124.06	124.94	112.22
	F_{11}(5D)	6	4.00	4.00	4.00	4.00	4.00	3.96
	F_{12}(20D)	8	0.02	0.02	0.02	0.00	0.00	0.00

TABLE X

AVERAGE NUMBER OF PEAKS FOUND BY BNDE WITH/WITHOUT DPO

Problem ID	NKP	BNDE 1.0E-01	1.0E-04	BNDE-nd 1.0E-01	1.0E-04
F_4(2D)	4	4.00	4.00	4.00	4.00
F_6(3D)	81	76.24	66.0	70.24	56.26
F_7(3D)	216	216.00	117.50	175.02	78.30
F_{11}(5D)	6	6.00	4.00	5.68	4.00
F_{12}(20D)	8	7.48	0.02	7.22	0.02

advantages of training NNEs by a niching algorithm and then present the experiments and results.

A. Methodology

Training a neural network is an optimization problem: the variables are the parameters involved in the network (e.g., the weights and biases of a multilayer perception), while the objective function is to minimize the sum of errors between the desired and practical output values. Fig. 10(a) shows a very simple neural network that contains only two weights w_1 and w_2. According to the different distributions of data, the solution landscape of w_1 and w_2 varies. A typical example is plotted in Fig. 10(b), which contains two peaks (they should be valleys actually, but we reverse the landscape for easy observation). Using a traditional gradient-based algorithm such as the BP for optimization, the weights will converge to one of the peaks, either the global or the local one. If we generalize the neural network to a more complicated one involving more neurons and connections, the solution space becomes higher dimensional, which could involve much more global and local optima. The BP algorithm may get trapped in a weak local optimum, resulting in unsatisfactory performance of the neural network. Besides, even in some cases the BP algorithm successfully finds the global optimum, the generalization ability of neural network may still be limited due to the difference between the training and the test data. Targeting at improving the overall performance, NNE trains a number of neural networks for the same task and aggregates the results. The idea behind NNE is that, using different training subsets and initial parameters, the optimization algorithm can detect different optima in the solution landscape. Then, combining a set of neural networks with diverse weights can significantly enhance the generalization ability.

From the above discussion, it can be noticed that the niching method (which is capable of learning multimodal parameters

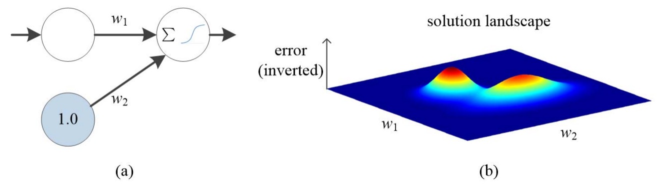


Fig. 10. Simple neural network. (a) Network structure. (b) Inverted solution landscape.

simultaneously) is very suitable to train NNEs. In addition, the use of a niching algorithm brings the following advantages. First, the traditional BP and EAs encounter difficulties in maintaining the diversity of NNEs. This is because, by tuning the training subsets and initial weights, we cannot guarantee that the parameters will eventually converge to different positions at different executions. In comparison, the niching algorithm explicitly maintains multiple optima that are different from each other, which naturally resolves the diversity problem. Second, also due to the above-mentioned problem, commonly a relatively large number of neural networks are trained by an NNE for the sake of improving diversity. However, many of the networks are redundant and ensembling all of them is not helpful [6]. Using a niching algorithm with a final peak identification procedure, it is now possible to self-adapt the number of neural networks according to the data and to avoid redundancy. Third, the traditional algorithms train a single neural network every time, which should be repeated a number of times by the NNE. Instead, the niching algorithm is able to optimize all components of the NNE in a single execution, which is more convenient.

In our implementation of applying BNDE for training NNEs, each individual is a floating-point vector that encodes the parameters involved in the neural network. Namely, each dimension denotes the weight of a connection. The individuals are randomly initialized and are then evolved according to the basic procedures of BNDE (**Algorithm 2**). In the fitness evaluation of an individual, we use the average error obtained by passing the training samples through the network of that individual. Finally, a set of solutions is obtained by performing the standard peak identification procedure on the combination of the final population and the archive. Each solution represents a different neural network in the NNE. Note that we need to make a slight amendment for the peak identification procedure in [46], since in practical applications the peak height value is unknown. In this paper, it is estimated by the global best fitness value found by the algorithm. Owing to the properties of BNDE, the base learners (namely, the neural networks) possess good diversity and accuracy, which meet the two essential requirements of ensemble learning.

B. Experiments

In order to validate the effectiveness of using BNDE for training NNEs, twelve well-known classification problems from the UCI machine learning repository [53] are tested. Table XI summarizes the properties of the data sets, including the number of samples, the number of features, and the number of classes. In the data preprocessing step, all features are constrained into range [0, 1] using min-max normalization. The tenfold cross-validation is performed.

TABLE XI

PROPERTIES OF THE 12 CLASSIFICATION DATA SETS

Datasets	No. samples	No. features	No. classes
Iris	150	4	3
Wine	178	13	3
Seeds	210	7	3
Heart	270	13	2
Parkinsons	195	22	2
Fertility	100	9	2
Climate	540	18	2
Diabetes	768	8	2
Balance	625	4	3
Ionosphere	351	34	2
Wisconsin	683	9	2
Segmentation	2310	19	7

TABLE XII

COMPARISONS OF THE ACCURACY MEASURES ON THE DATA SETS (MEAN ± STANDARD DEVIATION)

Datasets	BP-NN	BP-NNE	BNDE-NNE
Iris	90.67 (± 10.52)	95.33 (± 5.49)	**96.67 (± 4.71)**
Wine	95.00 (± 6.11)	98.33 (± 2.68)	**98.89 (± 2.34)**
Seeds	95.71 (± 4.17)	**96.67 (± 3.21)**	96.19 (± 3.76)
Heart	80.37 (± 7.42)	82.59 (± 6.54)	**85.19 (± 7.20)**
Parkinsons	81.50 (± 9.14)	82.00 (± 6.32)	**86.50 (± 5.80)**
Fertility	**91.00 (± 7.38)**	81.00 (± 12.87)	85.00 (± 8.50)
Climate	91.11 (± 4.35)	90.74 (± 4.00)	**94.63 (± 2.95)**
Diabetes	75.97 (± 3.01)	74.29 (± 4.69)	**80.39 (± 4.76)**
Balance	90.32 (± 3.70)	**92.86 (± 3.98)**	88.25 (± 4.97)
Ionosphere	74.44 (± 7.27)	88.06 (± 6.01)	**92.22 (± 4.86)**
Wisconsin	85.65 (± 14.47)	93.62 (± 11.19)	**96.38 (± 2.67)**
Segmentation	79.39 (± 3.97)	**83.55 (± 2.30)**	81.52 (± 5.70)

In the comparison, three methods are tested, all based on an identical neural network with one hidden layer of five neurons. The basic parameters such as the learning rate and momentum are also kept identical. Note that we do not fine-tune the network architecture and control parameters to improve the absolute performance. We care more about the relative performance, in order to see the effectiveness of using the niching method for ensemble learning. BP-NN indicates the single neural network using BP for training, while BP-NNE represents the ensemble of BP-NNs. The number of networks in BP-NNE is set to 20 for good performance. Then, BNDE-NNE denotes the NNE optimized by our proposed BNDE algorithm. The algorithm is able to self-adjust the number of networks in use. However, to avoid optima explosion, we limit the number of networks produced by BNDE-NNE by an upper bound of 30. Finally, the prediction results of neural networks are combined by the majority voting.

Table XII reports the accuracy measures obtained by the methods on different data sets, where the best results are marked in bold. Particularly, the percentage of improvements made by BNDE-NNE against BP-NN and BP-NNE is plotted, respectively, in Fig. 11. It can be observed that BNDE-NNE achieves the best results on eight out of the twelve data sets. Meanwhile, the magnitude of improvements achieved by BNDE-NNE is considerable in most cases.

In addition, we present the average number of neural networks produced by BNDE-NNE on different data sets in Fig. 12, from which a few interesting findings can be obtained. First and the most intuitively, the results in Fig. 12 indicate that, for different data sets, the optimal number of neural

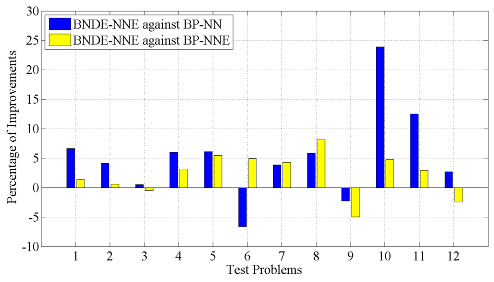


Fig. 11. Percentage of improvements made by BNDE-NNE against BP-NN and BP-NNE (numbers in the horizontal axis indicate the different data sets: 1-Iris, 2-Wine, 3-Seeds, 4-Heart, 5-Parkinsons, 6-Fertility, 7-Climate, 8-Diabetes, 9-Balance, 10-Ionosphere, 11-Wisconsin, and 12-Segmentation).

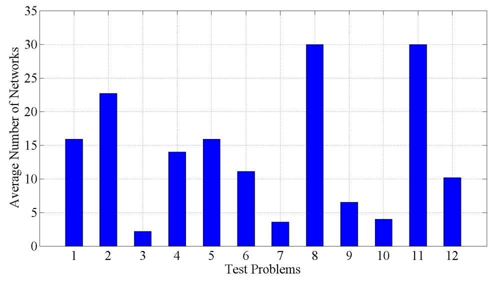


Fig. 12. Average number of neural networks produced by BNDE-NNE (numbers in the horizontal axis indicate the different data sets: 1-Iris, 2-Wine, 3-Seeds, 4-Heart, 5-Parkinsons, 6-Fertility, 7-Climate, 8-Diabetes, 9-Balance, 10-Ionosphere, 11-Wisconsin, and 12-Segmentation).

networks in NNEs differ a lot. Using a fixed number of base learners for all data sets may limit the performance of ensemble learning. Second, actually our results reveal some hidden properties of the solution landscapes for different data sets. For example, the "Diabetes" and "Wisconsin" data sets involve lots of optima in the solution landscapes. On the contrary, the number of optima for the "Seeds," "Climate," and "Ionosphere" data sets are much less (less than 5 as estimated by our proposed method). For this type of data sets, the traditional ensemble methods will incur many redundant base learners, which destroy the balance of different effective learners. Third, these findings enlighten us on a new application direction of using niching algorithms to assist the machine learning paradigms. Instead of optimizing the parameters, the niching algorithms can be utilized as a preprocessing step to analysis the problem models or data structures, so as to guide the design or control of the machine learning algorithms.

VII. Conclusion

This paper explored the new possibility of learning multimodal parameters for machine learning models. First, we pointed out four potential working scenarios where multiple sets of effective parameters are useful: 1) generating base learners in ensemble learning; 2) introducing posterior user preferences; 3) providing geometric interpretation for the problem; and 4) tackling physical constraints and system dynamics. To complete the task of learning multimodal parameters, multimodal optimizers are required. We then developed an efficient neighborhood-based niching algorithm named BNDE for MMOP. Using GBDE as the baseline algorithm, BNDE is free from tuning the basic control parameters of DE. In order to locate multiple optima, the proposed algorithm performs local Gaussian reproduction in the neighborhoods, which facilitates fast contour matching. The index-based neighborhood strategy eliminates extra computational costs and the niche radius parameter. We further proposed and embedded a diversity-preserving operation termed DPO for the algorithm, which relieves inefficient search efforts of the population and then utilizes the saved computational costs for enhancing diversity. The experimental results validated that BNDE outperformed the others on standard MMOP problems. Moreover, the time efficiency of the proposed algorithm is much higher than the other competitive algorithms. Owing to the good performance in diversity and accuracy, the algorithm was applied to train NNEs. The advantages were validated through both empirical analyses and experimental simulations.

We summarize the potential future work into the following aspects.

1) The performance of BNDE decreases with the increasing requirement of accuracy to some extent, which indicates that its local exploitation ability still has room to improve. It would be helpful to adopt some methods, such as gradient-based local search, to enhance the exploitation ability.
2) It is desired and would be interesting to apply the algorithm to many other machine learning-related fields such as feature extraction, clustering, and landscape analysis.

References

[1] T.-C. Lu, G.-R. Yu, and J.-C. Juang, "Quantum-based algorithm for optimizing artificial neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 8, pp. 1266–1278, Aug. 2013.

[2] K. Dahal, K. Almejalli, M. A. Hossain, and W. Chen, "GA-based learning for rule identification in fuzzy neural networks," *Appl. Soft Comput.*, vol. 35, pp. 605–617, Oct. 2015.

[3] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multiobjective sparse feature learning model for deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3263–3277, Dec. 2015.

[4] Y. Ding, L. Cheng, W. Pedrycz, and K. Hao, "Global nonlinear kernel prediction for large data set with a particle swarm-optimized interval support vector regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2521–2534, Oct. 2015.

[5] R. Sun, F. Tsung, and L. Qu, "Evolving kernel principal component analysis for fault diagnosis," *Comput. Ind. Eng.*, vol. 53, no. 2, pp. 361–371, Sep. 2007.

[6] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artif. Intell.*, vol. 137, nos. 1–2, pp. 239–263, May 2002.

[7] T. Yu, J. Luo, and N. Ahuja, "Shape regularized active contour using iterative global search and local optimization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. Jun. 2005, pp. 655–662.

[8] E. Padilla-Rodal, O. Castanos, R. Bijker, and A. Galindo-Uribarri, "IBM-2 configuration mixing and its geometric interpretation for germanium isotopes," *Sociedad Mexicana de Fýsica AC*, vol. 52, no. 1, pp. 57–62, 2006.

[9] E. Dilettoso and N. Salerno, "A self-adaptive niching genetic algorithm for multimodal optimization of electromagnetic devices," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 1203–1206, Apr. 2006.

[10] Q. Yang, W.-N. Chen, Y. Li, C. P. Chen, X.-M. Xu, and J. Zhang, "Multimodal estimation of distribution algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 636–650, Mar. 2016.

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

[12] W. Dong and M. Zhou, "Gaussian classifier-based evolutionary strategy for multimodal optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1200–1216, Jun. 2014.

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

[14] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Jun. 2004, pp. 1382–1389.

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

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

[17] Y. Wang, H.-X. Li, G. G. Yen, and W. Song, "MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 830–843, Apr. 2015.

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

[19] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp.*, Apr. 2003, pp. 80–87.

[20] R. Poli and W. B. Langdon, "Markov chain models of bare-bones particle swarm optimizers," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2007, pp. 142–149.

[21] M. Campos, R. A. Krohling, and I. Enriquez, "Bare bones particle swarm optimization with scale matrix adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1567–1578, Sep. 2014.

[22] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 128–139, Jul. 2009.

[23] D. S. Yeung, J.-C. Li, W. W. Y. Ng, and P. P. Chan, "MLPNN training via a multiobjective optimization of training error and stochastic sensitivity," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 978–992, May 2016.

[24] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.

[25] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[26] R. A. Aliev, B. G. Guirimov, B. Fazlollahi, and R. R. Aliev, "Evolutionary algorithm-based learning of fuzzy neural networks. Part 2: Recurrent fuzzy neural networks," *Fuzzy Sets Syst.*, vol. 160, no. 17, pp. 2553–2566, Sep. 2009.

[27] H.-G. Han, W. Lu, Y. Hou, and J.-F. Qiao, "An adaptive-PSO-based self-organizing RBF neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.

[28] H.-M. Feng, "Self-generation RBFNs using evolutional PSO learning," *Neurocomputing*, vol. 70, nos. 1–3, pp. 241–251, Dec. 2006.

[29] M. You and T. Jiang, "New method for target identification in a foliage environment using selected bispectra and chaos particle swarm optimisation-based support vector machine," *IET Signal Process.*, vol. 8, no. 1, pp. 76–84, 2014.

[30] J. Zhang *et al.*, "Evolutionary computation meets machine learning: A survey," *IEEE Comput. Intell. Mag.*, vol. 6, no. 4, pp. 68–75, Nov. 2011.

[31] Z. Yan and J. Wang, "Nonlinear model predictive control based on collective neurodynamic optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 4, pp. 840–850, Apr. 2015.

[32] Q. Liu and J. Wang, "A one-layer projection neural network for nonsmooth optimization subject to linear equalities and bound constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 5, pp. 812–824, May 2013.

[33] C. Castillo, G. Nitschke, and A. Engelbrecht, "Niche particle swarm optimization for neural network ensembles," in *Advances in Artificial Life. Darwin Meets von Neumann*. Berlin, Germany: Springer, 2011, pp. 399–407.

[34] S. Kamyab and M. Eftekhari, "Feature selection using multimodal optimization techniques," *Neurocomputing*, vol. 171, pp. 586–597, Jan. 2016.

[35] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 189–201, Feb. 2009.

[36] M. Ware, E. Frank, G. Holmes, M. Hall, and I. H. Witten, "Interactive machine learning: Letting users build classifiers," *Int. J. Human-Comput. Stud.*, vol. 55, no. 3, pp. 281–292, Sep. 2001.

[37] H.-L. Ling, J.-S. Wu, Y. Zhou, and W.-S. Zheng, "How many clusters? A robust PSO-based local density model," *Neurocomputing*, vol. 207, pp. 264–275, Sep. 2016.

[38] W. Sheng, S. Chen, M. Sheng, G. Xiao, J. Mao, and Y. Zheng, "Adaptive multisubpopulation competition and multiniche crowding-based memetic algorithm for automatic data clustering," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 838–858, Dec. 2016.

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

[40] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahat, "Multimodal optimization using niching differential evolution with index-based neighborhoods," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.

[41] 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.

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

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

[44] 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.

[45] F. Pukelsheim, "The three sigma rule," *Amer. Statist.*, vol. 48, no. 2, pp. 88–91, 1994.

[46] X. Li, A. Engelbrecht, and M. Epitropakis, "Benchmark functions for CEC 2013 special session and competition on niching methods for multimodal function optimization," Evol. Comput. Mach. Learn. Group, RMIT Univ., Melbourne, VIC, Australia, Tech. Rep., 2013.

[47] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Finding multiple global optima exploiting differential evolution's niching capability," in *Proc. IEEE Symp. Differ. Evol.*, Apr. 2011, pp. 1–8.

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

[49] X. Li, "A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio," in *Proc. Genet. Evol. Comput. Conf.*, Jul. 2007, pp. 78–85.

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

[51] Y. Xiao, H. Wang, and W. Xu, "Hyperparameter selection for Gaussian process one-class classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 2182–2187, Sep. 2015.

[52] Y. Pan, R. Xia, J. Yin, and N. Liu, "A divide-and-conquer method for scalable robust multitask learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3163–3175, Dec. 2015.

[53] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://archive.ics.uci.edu/ml/

Yue-Jiao Gong (M'15) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

She was a Post-Doctoral Research Fellow with the Department of Computer and Information Science, University of Macau, Macau, China, in 2015 and 2016. She is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. She has authored over 50 papers in her research area. Her current research interests include evolutionary computation and machine learning methods as well as their applications to big data and intelligent transportation scheduling.

Dr. Gong currently serves as a reviewer for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON NEURAL NETWORK AND LEARNING SYSTEMS, and the IEEE TRANSACTIONS ON CYBERNETICS.

Jun Zhang (F'16) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Changjiang Chair Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. He has authored over 200 technical papers in his research area. His current research interests include computational intelligence, cloud computing, data mining, and power electronic circuits.

Dr. Zhang was a recipient of 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 INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS.

Yicong Zhou (SM'14) received the B.S. degree in electrical engineering from Hunan University, Changsha, China, and the M.S. and Ph.D. degrees in electrical engineering from Tufts University, Medford, MA, USA.

He is currently an Associate Professor and the Director of the Vision and Image Processing Laboratory with the Department of Computer and Information Science, University of Macau, Macau, China. His current research interests include chaotic systems, multimedia security, image processing and understanding, and machine learning.

Dr. Zhou was a recipient of the Third Price of Macau Natural Science Award in 2014. He serves as an Associate Editor of *Neurocomputing* and the *Journal of Visual Communication and Image Representation* and a Leading Co-Chair of Technical Committee on Cognitive Computing in the IEEE Systems, Man, and Cybernetics Society.