Neurocomputing 149 (2015) 29–38

Contents lists available at ScienceDirect

Neurocomputing

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

Evolved neural network ensemble by multiple heterogeneous swarm intelligence

Zeng-Shun Zhao [a,c,*], Xiang Feng [a], Yan-yan Lin [a], Fang Wei [a], Shi-Ku Wang [a], Tong-Lu Xiao [a], Mao-Yong Cao [a], Zeng-Guang Hou [b]

[a] College of Electronics, Communication and Physics, Shandong University of Science and Technology, Qingdao 266590, PR China
[b] State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, PR China
[c] School of Control Science and Engineering, Shandong University, Jinan, 250061, China

ARTICLE INFO

Article history:
Received 30 June 2013
Received in revised form
30 November 2013
Accepted 23 December 2013
Available online 1 August 2014

Keywords:
Neural network ensemble
Particle swarm optimization
Differential evolution
Artificial bee colony
Chaotic search

ABSTRACT

The neural network ensemble (NNE) is a very effective way to obtain a good prediction performance by combining the outputs of several independently trained neural networks. Swarm intelligence is applied here to model the population of interacting agents or swarms that are able to self-organize. In this paper, we combine NNE and multi-population swarm intelligence to construct our improved neural network ensemble (INNE). First, each component forward neural network (FNN) is optimized by chaotic particle swarm optimization (CPSO) and gradient gescending (GD) algorithm. Second, in contrast to most existing NNE training algorithm, we adopt multiple obviously different populations to construct swarm intelligence. As an example, one population is trained by particle swarm optimization (PSO) and the others are trained by differential evolution (DE) or artificial bee colony algorithm (ABC). The ensemble weights are trained by multi-population co-evolution PSO–ABC–DE chaotic searching algorithm (M-PSO–ABC–DE–CS). Our experiments demonstrate that the proposed novel INNE algorithm is superior to existing popular NNE in function prediction.

1. Introduction

Much work has been devoted to finding the optimal or near optimal neural network architecture [1–5]. However, many real world problems are too complicated for any single neural network to deal in practice. There are many successesful instances from both natural and artificial systems to support that an integrated system consisting of several subsystems can fulfil a difficult duty satisfactorily. The development of multiple classifier systems (MCS) has gained lots of success as such systems can be more robust [3]. The success of neural network ensembles (NNE) in improving the generalization of the classifiers is a typical example [1,2].

Neural network ensemble (NNE) is a learning mechanism which has a collection of a finite number of neural networks trained for the same task. It was first proposed in Hansen and Salamon's work [6]. Its main idea is that the predicting ability of a neural network system could be significantly improved by assembling a set of neural networks, for example, training many neural networks and then combining their predictions in some way [2,7]. Neural network ensembles (NNE) are more robust than a single neural network, and can show graceful performance in situations where only a subset of neural networks can accomplish correctly [8]. There have been many evidences to support that ensembles generalize better than any single individuals in the ensemble [2]. It is wise to exploit the whole ensemble, rather than just the "best" single neural network.

However, how to construct such a neural network ensembles is still one open problem. One simple way to utilize the population information is to combine all individuals into an integrated system [2]. But the combined prediction would not be effective only by averaging, because in some cases, some components may behave unsatisfactorily. Various algorithms could be used to generate weights for such combination. In [9], the authors thought that it might be better to ensemble partial components other than all the trained neural networks; they introduced Genetic algorithm based selective ensembles (GASEN) to evolve the weights assigned to each forward neural network (FNN) for the best appropriate prediction. The binary particle swarm optimization (BiPSO) put forward by Kennedy and Eberhart in [10] could be utilized to optimize the NNE. In the BiPSO, the weight assigned to each individual FNN could be 0 or 1; and the ensemble problem of NNE is transferred into selecting several better appropriate FNNs by PSO. Another version of PSO, denoted as DePSO, is a little different from BiPSO, in which the weight of each individual FNN could be

* Corresponding author at: College of Electronics, Communication and Physics, Shandong University of Science and Technology, Qingdao 266590, PR China.
E-mail address: zhaozengshun@gmail.com (Z.-S. Zhao).

http://dx.doi.org/10.1016/j.neucom.2013.12.062
0925-2312/

decimal number. Minku [11] introduced an approach by explicitly partitioning the input variable space through a clustering method, which allowed a reduction in the number of nodes of the neural networks which compose the ensemble, thus reducing the execution time of the learning process. But how to determine the clustering parameters remains a problem. In [12], the authors Soares S. compared genetic algorithm and simulated snnealing based approaches for NNE. Their work aimed at selecting the best subset of models to be aggregated taking into account. Thus the diversity of whole populations may be weakened largely. In the literature [8], the authors presented the regularized negative correlation learning algorithm for the whole ensemble. Adding a correlation penalty term to the cost function of each individual network, each neural network could minimize its mean square error together with the correlation of the ensemble. In [1], Symone Soaresa introduced genetic algorithm to select both the best subset of FNN and the optimal combination strategy to ensure the accuracy and the robustness of the ensemble.

The FNN often adopts GD algorithm to optimize its parameters since GD could achieve higher convergent accuracy and faster convergent speed around the global optimum. But it should be noticed that the GD algorithm has its own disadvantage: it would easily get trapped in its local minima, especially for the non-linear pattern classification problems or complex function approximation problem. As a result, GD may lead to a failure in finding a global optimal solution [13]. The PSO algorithm is shown to converge rapidly during the initial stages of a global search. But around global optimum, the search process may become very slow, which makes the improvement decrease gradually with more searching iterations [14,15]. In [16,17], authors utilized DE to optimize PSO to improve the efficiency and precision.

In [18], the authors proposed to couple PSO and BP to train the weights of FNN, where the hybrid algorithm could make use of both global search ability of the PSO and local search ability of the BP algorithm. The authors in [19] presented the tent mapping chaotic PSO to avoid trapping into local minima and improve the performance of neural network for predictive control. The authors in [20] also put forward an improved chaotic PSO which applied the chaotic mapping and the adaptive inertia weight factor to enhance the precision of PSO. The authors in [21] proposed a hybrid method which combined the fuzzy set theory and PSO–BP algorithm to determine the weight of the component FNNs, then utilized the weights to synthesize their assessment result to form the final output.

In this paper, we concentrated on introducing a novel evolved neural network ensemble (NNE) framework, which incorporated the multiple evolution algorithms, would be more superior than the traditional NNE. In other words, our purpose is that optimizing the weights of the NNE could largely improve the prediction of NNE. Actually, all kinds of the evolution algorithm can be integrated in such a framework, and their differences only lie in the respective evolving types. In this paper, we implement PSO, DE and ABC as the concrete cases of the multiple heterogeneous evolution algorithms. PSO, DE and ABC are similar to the Genetic Algorithm (GA) and Ant Colony (AC) in the sense that these two evolutionary heuristics are population-based search methods, but the algorithms we selected are more computationally efficient than the rest. Another desirable characteristic is that PSO, DE and ABC require minimal parameters to tune.

Hence, in our paper, we propose the improved CPSO–GD–NNE mode, where we couple the chaotic PSO (CPSO) and GD algorithm to train each component FNN, and then use the multiple population co-evolution to optimize the weights of NNE. There are two stages: In the FNN training stage, we firstly use CPSO to train each component FNN to quickly find the global minimum yet the coarse solutions. When the constriction condition is satisfied, we apply the GD algorithm until the new termination condition reached. In the NNE training stage, we integrate chaotic searching into multiple population co-evolution algorithm, which uses PSO, DE and Artificial bee colony (ABC), and present the multi-population co-evolution chaotic search (M-PSO–ABC–DE–CS) to train weights of the ensemble.

This paper is organized as follows: Section 2 briefly describes the principle of GD algorithm, PSO, chaotic mapping and the CPSO–GD couple algorithms. Section 3 outlines the principle of DE, ABC and the multi-population co-evolution algorithm (M-PSO–ABC–DE–CS). The simulation results and comparison are presented in Section 4. Finally, Section 5 provides some concluding remarks.

2. Component neural network optimized by CPSO and GD algorithms

2.1. Principle of gradient descending algorithm

The gradient descending technique proposed by Werbos [22], is widely used in optimizing and training neural networks [4,5]. But its disadvantage which is sensitive to the initial weight vector, often leads to a different result by virtue of different weight vector. This disadvantage makes troubles in trapping in a local solution which is bias to the best solution, but it could achieve faster convergent speed around global optimum. Due to the reasons above, we couple the CPSO and GD algorithm to optimize the FNN. The detailed gradient descending technique is described in [22,23].

If we suppose that $X=\{x_1,x_2\cdots x_m\}$ has m input samples and $Y=\{y_1,y_2\cdots y_n\}$ has n output results. There are p neurons in hidden-layer and q neurons in output-layer. The thresholds of hidden neurons are $\theta^1=\{\theta_1^1,\theta_2^1\cdots\theta_p^1\}$ and thresholds of the output-layer are $\theta^2=\{\theta_1^2,\theta_2^2\cdots\theta_q^2\}$. We represent the weights between input-layer and hidden-layer as $V=\{v_{11},v_{12}\cdots v_{21}\cdots v_{np}\}$, weights between hidden-layer and output-layer as $W=\{w_{11},w_{12}\cdots w_{21}\cdots w_{pq}\}$. The transition function of hidden-layer is $f(\cdot)$ and the transition function of output-layer is $g(\cdot)$ (Fig. 1).

Consider the error between final output and the expected output is E given by

$$E=\frac{1}{2}\sum_{k=1}^{n}\{y_k-g[\sum_{j=1}^{p}w_{kj}f(\sum_{i=1}^{m}v_{ij}x_i-\theta_j^1)-\theta_k^2]\}^2 \tag{1}$$

We could get the updating formula of the two weights, as follows:

$$v_{ji}(t+1)=v_{ji}(t)+\Delta v_{ji}=v_{ji}(t)-\eta^1\frac{\partial E}{\partial v_{ji}} \tag{2}$$

$$w_{kj}(t+1)=w_{kj}(t)+\Delta w_{kj}=w_{kj}(t)-\eta^2\frac{\partial E}{\partial w_{kj}} \tag{3}$$

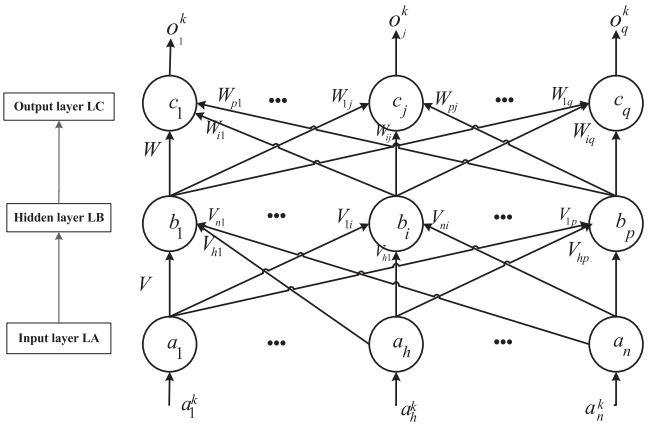


Fig. 1. The forward neural network architecture.

The two thresholds can be updated as follows,

$$\theta_j^1(t+1)=\theta_j^1(t)+\Delta\theta_j^1=\theta_j^1(t)+\eta^1\frac{\partial E}{\partial\theta_j^1} \tag{4}$$

$$\theta_k^2(t+1)=\theta_k^2(t)+\Delta\theta_k^2=\theta_k^2(t)+\eta^2\frac{\partial E}{\partial\theta_k^2} \tag{5}$$

where η^1,η^2 is the learning rate. But it should be noticed that the learning rate, which controls the convergence to a local optimal solution, is often determined by experiments or experience. If it is not ideal enough, it would easily result in oscillating of the network and could not converge rapidly.

2.2. Particle swarm optimization algorithm

The PSO algorithm could be described as a swarm of birds or pigeons hovering in the sky for food. We assume the pigeon swarm as a random particle swarm, each particle as one bird. Each bird has its own location and flying velocity. One n dimensional swarm with m particles is denoted as $X_i=(x_{i1},x_{i2}\cdots x_{in})$ and $V_i=(v_{i1},v_{i2}\cdots v_{in})$ where X_i and V_i are the position and velocity of the ith particle in n dimensional space. At each step of iterations, the particles adjust their trajectories and velocities according to their own previous best value $P_i=(p_{i1},p_{i2}\cdots p_{in})$ and the best global value of all particles in the population $G_i=(g_1,g_2\cdots g_n)$ up to the current step. After having obtained the two best values, each particle updates its position and velocity according to the following rules:

$$v_{i,d}^{k+1}=w\times v_{i,d}^k+c_1\times r_1\times(pbest_{i,d}^k-x_{i,d}^k)+c_2\times r_2\times(gbest_d^k-x_{i,d}^k) \tag{6}$$

$$x_{i,d}^{k+1}=x_{i,d}^k+v_{i,d}^{k+1} \tag{7}$$

Here, r_1 and r_2 are two uniformly distributed random numbers in the interval of [0, 1]. In the general PSO, c1 and c2 are the cognitive and social learning factor respectively. They could determine the magnitude of the random forces in the direction of the personal best $pbest_i$ and the swarm best $gbest$, usually $c_1=2.8$, $c_2=1.3$. w^t is the inertial weight used to balance the global and local searching. The detailed description could be referred in [24,25].

2.3. Chaotic local searching

Chaos is a bounded unstable dynamic behavior that exhibits sensitive dependence on initial conditions and includes infinite unstable periodic motions. Although it appears to be stochastic, it occurs in a deterministic non-linear system under deterministic conditions [20]. By following chaotic ergodic orbits, the chaotic searching could traverse every state in a certain region by its own regularity. Especially every state could be visited only once.

Due to its easy implementation and excellent property to avoid trapping in local optima, in our paper, we employ the logistic mapping to construct our improved chaotic PSO algorithm where the chaotic dynamic searching would enhance the PSO global optimization ability.

The logistic equation is defined as follows:

$$z_{n+1}=\mu\times z_n(1-z_n),\quad 0\le z_0\le 1 \tag{8}$$

where μ is the control parameter. z_n indicates the nth iteration decision variable. The value of μ and z_0 are defined by experiments.

We obtain the chaotic variable by the following equation, which could travel over the whole search space using the property of chaos ergodicity.

$$s_j^k=\frac{x_j^k-x_{\min\ j}}{x_j^k-x_{\max\ j}},\quad j=1,2,\ldots,m \tag{9}$$

Here s_j^k and x_j^k denote the respecive chaotic variable and the decision variable of the jth dimension in the kth iteration. What's more, $0\le s_j^k\le 1$. $x_{\min\ j}$ and $x_{\max\ j}$ represent the lower and the upper bound value of the jth dimension.

The chaotic variable is updated according to the following logistic equation:

$$s_j^{k+1}=4\times s_j^k(1-s_j^k),\quad j=1,2,\ldots,m \tag{10}$$

Next, we would convert the chaotic variables s_j^k to decision variable x_j^k by the following equation:

$$x_j^{k+1}=x_{\min\ j}+s_j^{k+1}(x_{\max\ j}-x_{\min\ j}),\quad j=1,2,\ldots,m \tag{11}$$

By virtue of above Eqs. (9)–(11), we could accomplish the chaotic local searching process.

2.4. Coupled CPSO–GD to optimize the component FNN

The CPSO is proposed by fusing the advantages of both PSO and chaotic search to avoid the pre-maturity and easily trapping in local optimum. In addition, the GD algorithm could further optimize the FNN which has been improved by CPSO. The procedures for coupled CPSO–GD algorithm could be summarized as follows:

Step 1: Initialize the swarm of PSO: For m particles, set the initial weight w, the learning factor c_1,c_2, the maximal iterative generations $T_{\max\ -pso}$ and $T_{\max\ -GD}$.
Step 2: Update each particles by Eqs. (6) and (7), and evaluate the fitness of each particle in PSO, compute the previous best value of the ith particle up to the current step $pbest_i$ and the global best value of all particles in the population $gbest$.
Step 3: Memorize the 30% excellent particles, then perform the chaotic local searching via Eqs. (9)–(11) for $MaxC$ iterations.
Step 4: Evaluate the novel solutions with decision variables and update the $gbest$ and $pbest_i$.
Step 5: Shrink the search space by the following equation, and then produce the residual 70% particles.

$$x_{\min\ j}=\max\{x_{\min\ j},x_{g,j}-r\times(x_{\max\ j}-x_{\min\ j})\},\quad 0<r<1$$

$$x_{\max\ j}=\min\{x_{\max\ j},x_{g,j}+r\times(x_{\max\ j}-x_{\min\ j})\},\quad 0<r<1$$

Here, $x_{g,j}$ represents the value of the $gbest$ in the jth dimension.
Step 6: Do Step 2–5 until $t>T_{\max\ -pso}$ or do Step 7 if the best position has not changed for several iterations.
Step 7: Do GD algorithm until$t>T_{\max\ -GD}$.
Step 8: *if* $fitness(gbest)<E$ (E is the given threshold)
Output the prediction and the best solution.
else Continue to do GD for several iterations.
Output the prediction and the best solution.
endif

3. Neural network ensemble optimized by multi-population co-evolution algorithm

3.1. Principle of neural network ensemble

Network ensemble is usually made up of a linear combination of several networks which have been trained by same data set (the actual sample used by each component to learn could be different). Especially, each network within the ensemble is endowed with a potentially different weight to form the output of the ensemble.

After having obtained each refined component FNN as described in Section 2, we would concentrate on how to combine the output of each component FNN.

$$\bar{f}(x) = \sum_{i=1}^{n} \hat{w}_i f_i(x) \tag{12}$$

$$\hat{w}_i = \frac{w_i}{\sum_{i=1}^{n} w_i}, 0 < w_i < 1 \tag{13}$$

where $f_i(x)$ represents the output of the ith FNN and $\hat{w}_i$ is the importance weight associated to the ith FNN. More detailed descriptions of network ensemble can be referred in [2,26,27].

Our idea is to apply multiple heterogeneous intelligent swarms to optimize $\hat{w}_i$ of each individual component network, hence to obtain the best appropriate prediction as formula (12). We define $\hat{w} = \{\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_n\}$ as one 'bird' of PSO, one 'chromosome' of DE, or one 'bee' of ABC. Moreover, given the testing condition or the objective requirement, the weights of NNE would be transferred to the optimization problem in the integration of PSO, DE or ABC.

3.2. Differential evolution algorithm

Here, we introduce another global searching algorithm, differential evolution algorithm [28]. DE is also a floating point encoded evolutionary algorithm for global optimization over continuous spaces, but it creates new candidate solutions by combining the parent individual and several other individuals of the same population. It consists of selection, crossover and mutation operations.

The procedures for DE algorithm are as following:

Step 1: Initialization

Initial population $\{x_i(0)|x_{j,i}^L \le x_{j,i}(0) \le x_{j,i}^U, \quad i = 1, 2, \ldots, NP, \quad j = 1, 2, \ldots, D\}$ (NP represents the size of population and D denotes dimensionality) is produced as below:

$$x_{j,i}(0) = x_{j,i}^L + r \times (x_{j,i}^U - x_{j,i}^L) \tag{14}$$

where $x_i(0)$ represents the ith individual and $x_{j,i}(0)$ represents the jth gene of the ith individual in initialized generation. r is the uniformly distributed random number in the range [0, 1]. x^L and x^U represent the lower and upper bound value of corresponding dimensionality.

Step 2: Mutation

DE realizes the individual mutation by differential strategy. Its main idea is to using the scaled differential vector to increase diversity into the new generation. The simple method is to choose two individuals randomly and obtain the mutated individual as below:

$$v_i(g+1) = x_{r1}(g) + F \times (x_{r2}(g) - x_{r3}(g)), \quad i \ne r1 \ne r2 \ne r3 \tag{15}$$

where $v_i(g)$ represents the ith mutated individual in the gth generation. Meanwhile $\{v_i(g+1)|v_{j,i}^L \le v_{j,i}(g) \le v_{j,i}^U, \quad i = 1, 2, \ldots, NP, \quad j = 1, 2, \ldots, D\}$, that is to say the boundary condition for each gene of the individual should be satisfied. F indicates a amplification factor.

Step 3: Crossover

Crossover operation is implemented between the gth individual $x_i(g)$ and mutated variable $v_i(g+1)$.

$$u_{j,i}(g+1) = \begin{cases} v_{j,i}(g+1), & \text{if } r \le CR \\ x_{j,i}(g), & \text{otherwise} \end{cases}, \tag{16}$$

where CR denotes the crossover probability, and r is the uniformly distributed random number in the range [0, 1]

Step 4: Selection

$$x_i(g+1) = \begin{cases} u_i(g+1), & \text{if } f(u_i(g+1)) \le f(x_i(g)) \\ x_i(g), & \text{otherwise} \end{cases} \tag{17}$$

where $x_i(g+1)$ is the offspring of the $(g+1)$th generation.

By virtue of Eqs. (14)–(17), we could accomplish the DE evolution process.

3.3. Artificial bee colony algorithm

As a swarm intelligence algorithm, the idea of artificial bee colony algorithm (ABC) is to imitate the behavior of honeybee swarms for foraging nectar source. The basic model of ABC consists of three essential groups: employed bees, onlookers and scouts. In a bee colony, half of them could be onlookers and the others are the employed bee. For each nectar source, there is one employed bee. In other words, the number of employed bee is equal to the number of food source [29]. The employed bee which has abandoned an over-exploiting nectar source could immediately be a scout. The employed bees could share their information with onlookers by waggle dancing and then the onlookers select one of the food sources. For more detailed descriptions of artificial bee colony algorithm are referred in [29–31].

The procedures for ABC algorithm are as following:

Step 1: Initialization

Initialize the population $\{X_1, X_2, \ldots, X_s\}, X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ represent not only an employed bee, but also the $i-th$ nectar source, which means the solution vector of the optimization problem in d dimensional space. s denotes the size of swarm.

$$x_{ij} = x_{ij}^{\min} + r \times (x_{ij}^{\max} - x_{ij}^{\min}) \tag{18}$$

where r is the uniformly distributed random number in the range [0, 1]. $x_{ij}^{\min}$ and $x_{ij}^{\max}$ represent the lower and upper bound value of corresponding dimensionality.

Step 2: Employed bee phase

Each employed bee performs local searching within its neighbor to forage a better nectar source by the following equation.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{19}$$

where v_{ij} represents an updated location by the local search mechanism and ϕ_{ij} is a random number which is used to control the search radius.

Then we calculate the fitness value of the new location and compare it with the previous one.

$$x_i(t+1) = \begin{cases} X_i(t), trial = trial + 1 & \text{if } fit(V_i) \le fit(X_i) \\ V_i(t), trial = 0 & \text{otherwise} \end{cases}, \tag{20}$$

The notion *trial* represents the frequency of exploiting in a same food source.

Step 3: Onlooker phase.

The employed bees store the food source information and share with onlookers by the waggle dances.

$$p_i = \frac{fit}{\sum_{j=1}^{s} fit_j}, \tag{21}$$

where P_i presents the probability value associated with the ith nectar source. The onlookers select the nectar source by the greedy selection mechanism according to the probability, which means that the new location with equal or better solution will replace the previous source [23].

Step 4: Scout phase
If *trial* > max *_trial*, it means that the current food source has been exhausted. The employed bee would be a scout and produce a new location by Eq. (18), then memorize the best nectar source so far.

The ABC evolution process would accomplish by virtue of Eqs (18)–(21).

3.4. Multiple population co-evolution to optimize neural network ensemble

In this paper, we introduce the multiple population co-evolution algorithm which could not only avoid trapping into the local solution, but also increase the diversity of the entire population. The cooperative algorithm tends to compensate for disadvantage of individual method and could be apt to the best solution. At the end of each iteration, we compare the offsprings of two or three populations optimized respectively by heterogeneous search algorithms, and select several better appropriate solutions which determines the evolution direction, finally execute the chaotic searching.

Here, we describe the architecture of multi-population co-evolution PSO–ABC–DE chaotic searching algorithm (M-PSO–ABC–DE–CS) and multi-population co-evolution PSO–ABC–DE algorithm (M-PSO–ABC–DE). In the architecture, we suppose that population A would be evolved by PSO, population B would be evolved by DE and population C would be evolved by ABC, moreover, each population has same size of solution vectors. Actually, all kinds of the evolution algorithm can be integrated in such a framework, and their differences only lie in the respective evolving types. That is to say, of course, here the evolution algorithm may be replaced by Genetic Algorithm (GA), Ant Colony (AC) or other kinds. In this paper, we implement PSO, DE and ABC as the concrete cases of the multiple heterogeneous evolution algorithms. PSO, DE and ABC are similar to the Genetic Algorithm (GA) and Ant Colony (AC) in the sense that these two evolutionary heuristics are population-based search methods, but the algorithms we selected are more computationally efficient than the rest. Another desirable characteristic is that PSO, DE and ABC require minimal parameters to tune.

Our architecture is as following (Figs. 2 and 3)

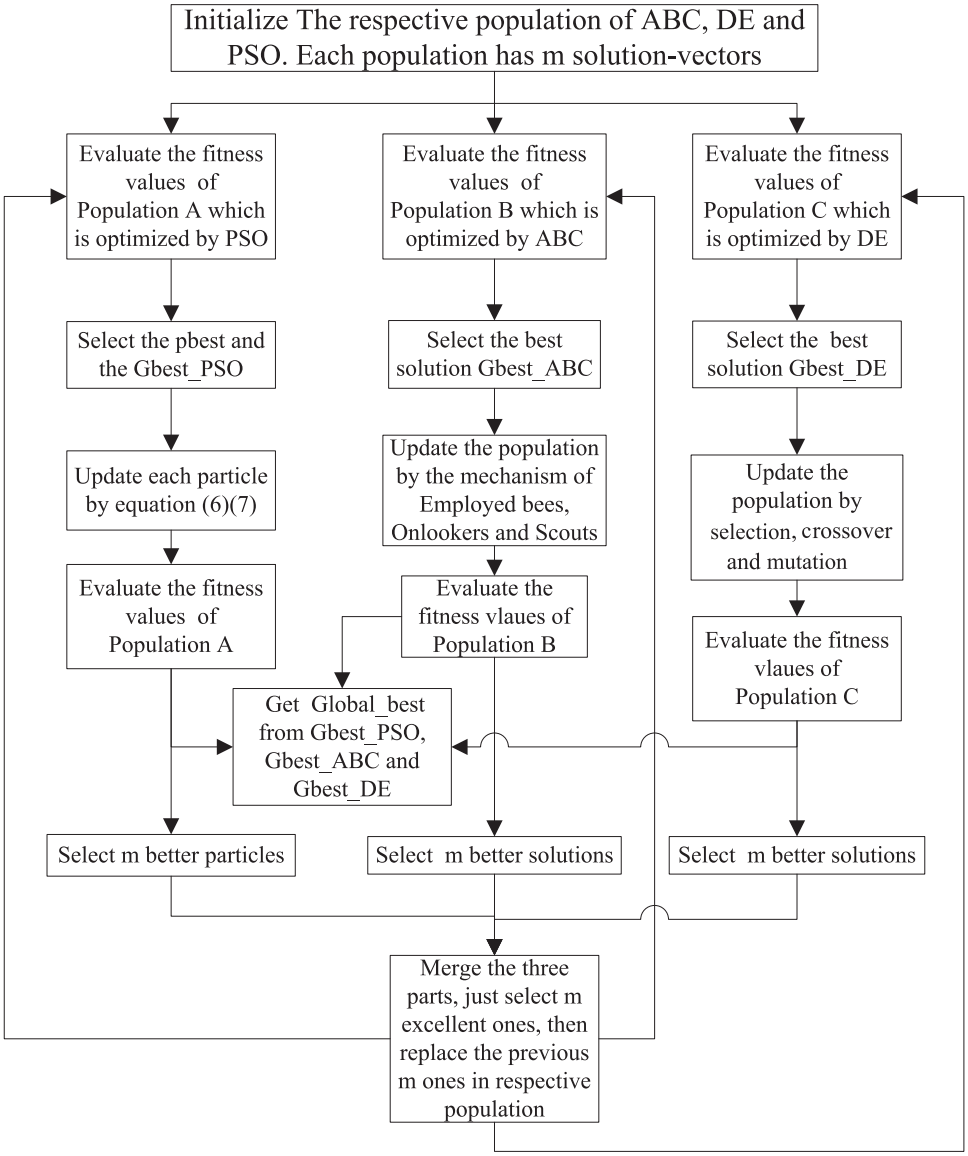


Fig. 2. M-PSO-ABC-DE mode of multiple population co-evolution optimized neural network ensemble.

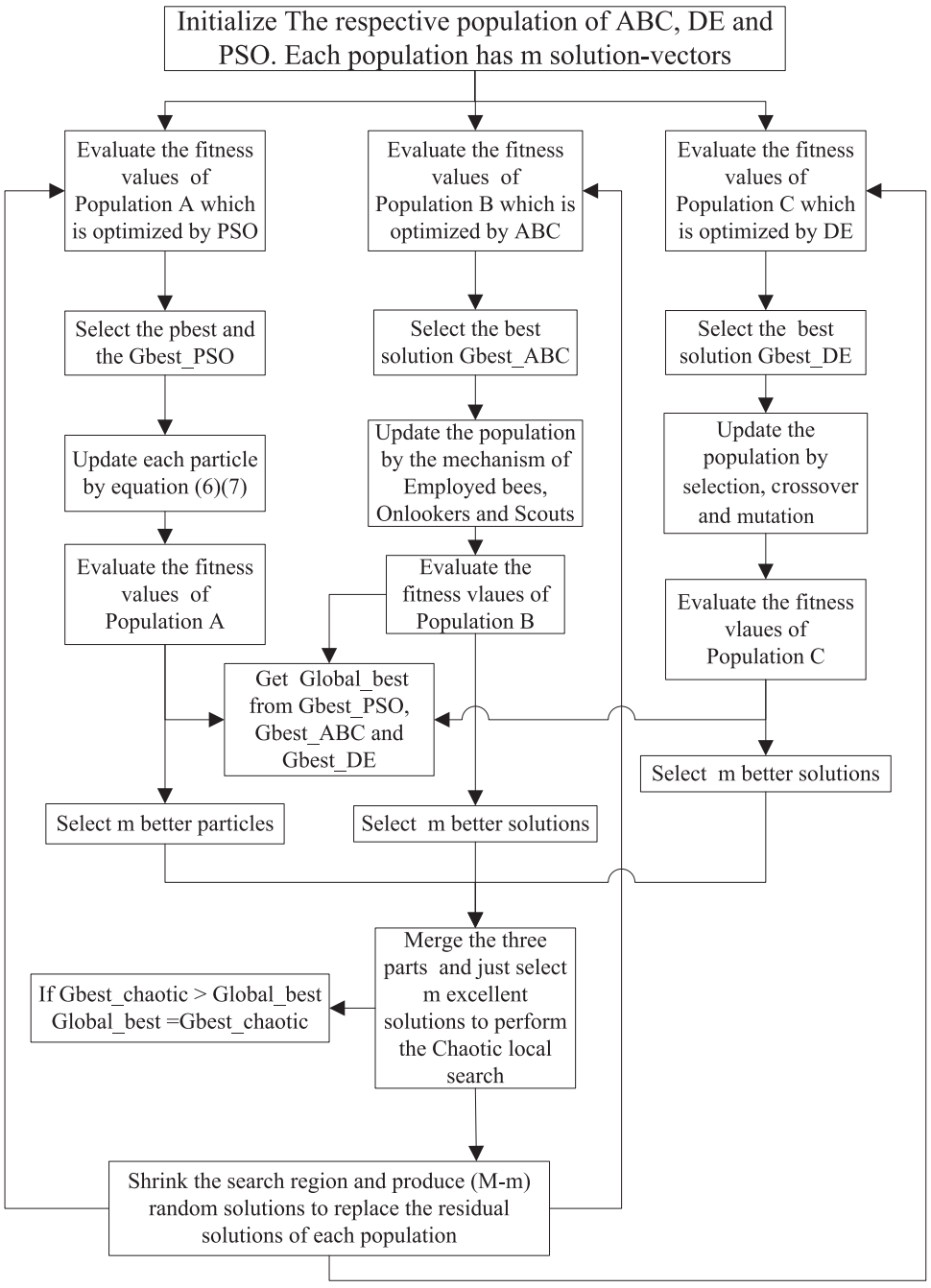


Fig. 3. M-PSO–ABC–DE–CS mode of multiple population co-evolution optimized neural network ensemble.

4. Experiments and results

In this section, we first compare the optimization performance of multiple heterogeneous swarm intelligence algorithm against single evolution algorithm over benchmark functions. Next, to test the efficiency of the improved NNE, we perform the comprehensive experiments to compare different NNE based on various optimization methods.

4.1. Optimization comparison on benchmark functions

To demonstrate the advantage of the multiple heterogeneous swarm intelligence algorithm, we have presented several algorithms for comparative experiments, such as artificial bee colony (ABC), standard PSO (SPSO), chaotic PSO (CPSO), differential evolution (DE), multi-population co-evolution SPSO (M-SPSO), multi-population co-evolution CPSO (M-CPSO), multi-population co-evolution DE (M-DE), multi-population co-evolution PSO–DE chaotic searching algorithm (M-PSODE–CS), multi-population co-evolution PSO–DE (M-PSODE), multi-population co-evolution PSO–ABC–DE chaotic searching algorithm (M-PSO–ABC–DE–CS) and multi-population co-evolution PSO–ABC–DE algorithm (M-PSO–ABC–DE). Each algorithm contains 30 particles or 30 solution-vectors in their population, and runs 100 iterations before termination. We select two well known benchmark function with multiple peaks, Ackley function and Rosenbrock function, to validate the proposed method.

Functioin #1 (Ackley function):$f(x_1,x_2)=-20\cdot e^{-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{2}x_i^2}}-e^{\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi\cdot x_i)}+20+e$, with $-30\le x_i\le 30$ and $e=2.71282$ with the global minimum at the orgin (0,0).

Functioin #2 (Rosenbrock function): $f(x_1,x_2)=(1-x_1)^2+100\times(x_2-x_1^2)^2$.

The optimization comparison of multiple heterogeneous swarm intelligence algorithm(in this case, denoted by M-PSO–DE–ABC–CS)against various swarm intelligence algorithms over Ackley function and Rosenbrock function, are summarized in Tables 1 and 2, respectively. The best solution in each test has been displayed in bold.

As shown in Table 1, in the optimization test of the multi-model Ackley functions, the proposed algorithm (M-PSO–DE–ABC–CS) and its variants, M-PSO–DE–ABC, M-PSO–DE–CS could always obtain the best solutions compared against the existing PSO, DE, CPSO etc.

Compared Table 1 with Table 2, for optimization on different benchmark function, all the algorithms exert their efforts, but each individual single algorithm reflect different predominance by reason of diversity in multiple peak function. For example, ABC and DE on Rosenbrck function outperform themselves on Ackley function. That is to say, each individual single algorithm with different evolution mechanism cannot perform consistently to handle various optimzation problem.

Table 1
The optimization comparison of swarm intelligence algorithms over Ackley function.

The optimized method	Functioin #1 (Ackley function)	
	Global optima point (0,0)	Global optimum value=0
SPSO	(0.0085146, −6.8746e−005)	0.026013
DE	(0.017979, 0.012996)	0.04182
ABC	(0.010271, 0.014724)	0.059339
CPSO	(−0.0015835, 0.00024026)	0.0045985
M-PSO	(−0.0012415, 0.0013141)	0.0052001
M-DE	(−0.0029124, 0.0025104)	0.0058888
M-ABC	(−0.0011618, −0.00032021)	0.0034473
M-CPSO	(−0.00038407, −0.00051755)	0.001834
M-PSO–DE–ABC–CS	(5.8284e−017, 2.3011e−016)	**−8.8818e−016**
M-PSO–DE–ABC	(4.0114e−016, −1.8302e−015)	2.6645e−015
M-PSO–DE	(−9.4369e−007, −7.9471e−007)	1.0168e−008
M-PSO–DE–CS	(1.8754e−017, −3.3154e−017)	2.1635e−009

Table 2
The optimization comparison of swarm intelligence algorithms over Rosenbrock function.

The optimized method	Functioin #2 (Rosenbrock function)	
	Global optima point (1,1)	Global optimum value=0
SPSO	(0.87654, 0.75262)	0.039932
DE	(0.65133,0.40074)	0.0050814
ABC	(0.7115, 0.44608)	0.0077588
CPSO	(0.89646, 0.79567)	0.017074
M-PSO	(0.92627, 0.85707)	0.0055185
M-DE	(0.78409, 0.54509)	0.0035202
M-ABC	(0.84923, 0.747)	0.0065982
M-CPSO	(0.89422, 0.78314)	0.0038398
M-PSO–DE–ABC–CS	(1, 1)	**4.1995e−024**
M-PSO–DE–ABC	(1, 1)	5.6554e−020
M-PSO–DE	(0.9998, 0.99961)	4.2936e−008
M-PSO–DE–CS	(1, 1)	2.1635e−009

The multiple heterogeneous swarm intelligence algorithm, in this case, denoted by M-PSO–DE–ABC–CS, outperforms all the other variants according to the quality of the best solution. The combination of these different intelligent optimization methods using various evolving manners tends to compensate for any deficiencies of the individual swarm. Within the multiple heterogeneous swarm intelligence, this idea could absorb the advantage and remedy the disadvantage of respective algorithm, so that the architecture having multiple heterogeneous swarms could get rid of the sub-optimum more easily and obtain the best result. It can perform more complex tasks than any of its component (i.e., PSO, DE, ABC). It is more robust than a single component swarm strategy, and can show graceful performance degradation in situations where single swarm cannot perform correctly.

4.2. Performance comparison for NNE

First, we define the mean square error (MSE) between the real output and the predicted output, as the evaluating function.

$$MSE=\frac{\sum_{j=1}^{m}(y^j-\sum_{i=1}^{n}w_i\hat{y}_i^j)^2}{m} \tag{22}$$

where y^j represents the real output of the jth sample, $\hat{y}_i^j$ denotes the predicted y^j of the ith component FNN. w_i denotes the weight of each component FNN. The less the MSE is, the better the precision of INNE is.

We select the single-dimensionality function $y=1.1\times(1-x+2\times x^2)\times e^{(-x^2/2)}$ as the benchmark function #3, then select 100 training samples from $x=\{-4:0.08:4\}$ and select 100 testing samples from $\hat{x}=\{-3.96:0.08:3.96\}$.

In addition, we also select the multi-dimensionality function $y=0.5+(\sin^2(\sqrt{x_1^2+x_2^2})-0.5)/((1+0.001(x_1^2+x_2^2))^2)$ which is the Schaffer's F6 function as the second benchmark function #4, then select 100 samples randomly from $x_i\in\{-4:4\}$ for training and 100 samples from $\hat{x}_i\in\{4:12\}$ for testing.

Further more, we suppose that the NNE is comprised of 6 basic neural networks, each basic FNN has 3 layers (input–hidden–output), and there are 4 neurons in hidden-layer. In the training of INNE, to manifest the principles of fairness, each algorithm would incorporate 100 particles or solution-vectors, for the multi-population co-evolution algorithm, each sub-population would incorporate 50 particles or solution-vectors.

The NNE within each component FNN optimized by GD is trained by different algorithms, and the comparison is displayed in Table 3. The train-MSE represents the mean square error of the train set and also test-MSE represents of the test set. The precision of train-MSE and test-MSE could manifest the predicting ability of neural network ensemble.

The NNE in which the component FNN optimized by CPSO is trained by different algorithms, and the comparison is displayed in Table 4.

The NNE in which the component FNN optimized by CPSO–GD is trained by different algorithms, and the comparison is displayed in Table 5.

In Table 3, the SAV algorithm results in the largest MSE, no matter the train-MSE or test-MSE of function #3 and function #4. This is to say only by averaging, the combined prediction would not be effective. In some case, some component of NNE may not behave satisfactorily.

In ABC algorithm, as shown in Fig. 4, when somewhere has been exploited for a long time, the bee-colony would migrate to another place randomly, which leads to the excessive diversity and could not converge to a global solution.

In SPSO algorithm, it is showed to be premature around global optimum, and the improvement decreasing gradually with the searching iterating. Compared with SPSO, the CPSO algorithm in which the chaotic searching could improve the global converging ability of the SPSO, and the result is more precise than SPSO as Fig. 4 shows.

Table 3
The train-MSE and test-MSE comparison between several ensemble ways with each FNN optimized by GD.

The optimized method of NNE	Function 3#		Function #4	
	The train-MSE	The test-MSE	The train-MSE	The test-MSE
SAV	0.4806	0.4789	0.1366	0.1370
ABC	0.2530	0.2550	0.1176	0.1173
SPSO	0.2154	0.2174	0.1067	0.1067
CPSO	0.1972	0.1985	0.0932	0.0939
DE	0.1292	0.1287	0.0925	0.0925
M-ABC	0.2217	0.2228	0.1052	0.1053
M-SPSO	0.2069	0.2087	0.0985	0.0992
M-CPSO	0.1772	0.1786	0.0898	0.0900
M-DE	0.1136	0.1121	0.0749	0.0755
M-PSO–DE–ABC–CS	0.0450	0.0449	0.0707	0.0709
M-PSO–DE–ABC	0.0457	0.0461	0.0716	0.0716
M-PSO–DE	0.0441	0.0445	0.0709	0.0711
M-PSO–DE–CS	0.0391	0.0394	0.0710	0.0712

Table 4
The train-MSE and test-MSE comparison between several ensemble ways with each FNN optimized by CPSO.

The optimized method of NNE	Function #3		Function #4	
	The train-MSE	The test-MSE	The train-MSE	The test-MSE
SAV	0.0311	0.0312	0.0625	0.0625
ABC	0.0165	0.0166	0.0500	0.0502
SPSO	0.0092	0.0086	0.0411	0.0415
CPSO	0.0077	0.0076	0.0405	0.0408
DE	0.0048	0.0046	0.0377	0.0339
M-ABC	0.0110	0.0109	0.0450	0.0450
M-SPSO	0.0052	0.0051	0.0395	0.0398
M-CPSO	0.0048	0.0047	0.0355	0.0357
M-DE	0.0044	0.0043	0.0354	0.0356
M-PSO–DE–ABC–CS	0.0028	0.0029	0.0335	0.0336
M-PSO–DE–ABC	0.0034	0.0034	0.0338	0.0339
M-PSO–DE	0.0033	0.0034	0.0337	0.0338
M-PSO–DE–CS	0.0030	0.0030	0.0340	0.0341

DE algorithm is famous for its ability to get rid of local solution by its mutation mechanism. In addition, we could see that for the single-algorithm to train, DE and CPSO could be more efficient than others. Last but not least, the multi-population co-evolution algorithms are all superior to their corresponding single-algorithms, especially our proposed algorithm M-PSO–ABC–DE–CS and M-PSO–DE–CS could be testified the best ones.

Compared with Table 3, each component FNN of Table 4 is optimized by CPSO and the result is obviously superior to Table 3. Because GD algorithm has its own disadvantage: it would easily get trapped in local optimization and lead to a failure in finding a global optimal solution. Whereas, chaotic searching could perform local searching as PSO traps in the local area. What's more, multi-population co-evolution algorithm could remedy the drawback which the single-algorithm could be apt to the local solution. Similar to Table 3, the multi-population co-evolution algorithms are all superior to their corresponding single-algorithms, especially our proposed algorithm M-PSO–ABC–DE–CS and M-PSO–DE–CS could be testified the best ones.

From Table 5, we could see that each component FNN optimized by CPSO and GD couple algorithm get the most excellent precision. Moreover, our proposed M-PSO–DE–CS also obtain the most accurate result. Compared with Tables 3 and 4, the M-PSO–DE–CS and M-PSO–ABC–DE–CS could converge to global solution

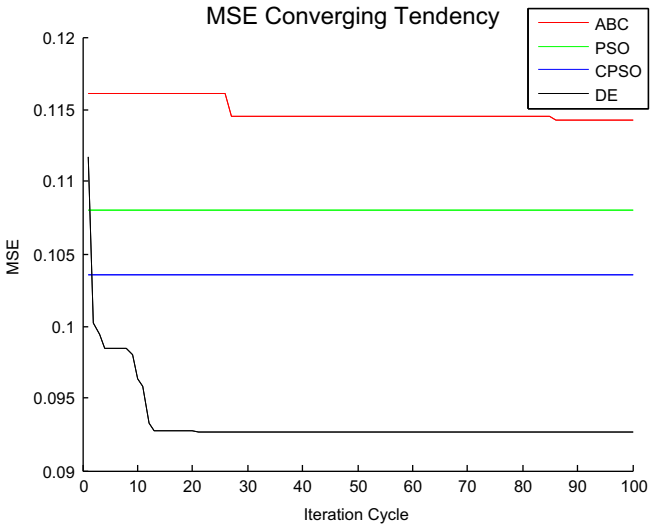


Fig. 4. MSE converging tendency of single-algorithm algorithm.

Table 5
The train-MSE and test-MSE comparison between several ensemble ways with each FNN optimized CPSO–GD couple algorithm.

The optimized method of NNE	Function #3		Function #4	
	The train-MSE	The test-MSE	The train-MSE	The test-MSE
SAV	3.3194e−007	3.3024e−007	2.5564e−004	2.5739e−004
ABC	3.2146e−007	3.2110e−007	2.5182e−004	2.5372e−004
SPSO	1.9895e−007	1.9723e−007	1.8568e−004	1.8817e−004
CPSO	1.3811e−007	1.3721e−007	1.7412e−004	1.7757e−004
DE	6.4246e−008	6.3243e−008	1.5211e−004	1.5263e−004
M-ABC	2.4132e−007	2.4073e−007	2.1193e−004	2.1402e−004
M-SPSO	1.3127e−007	1.2790e−007	1.6549e−004	1.6511e−004
M-CPSO	8.0708e−008	8.1377e−008	1.6016e−004	1.6189e−004
M-DE	5.0594e−008	5.0650e−008	1.4775e−004	1.4871e−004
M-PSO–DE–ABC–CS	3.0218e−008	3.0420e−008	1.3447e−004	1.3456e−004
M-PSO–DE–ABC	3.4422e−008	3.4237e−008	1.3662e−004	1.3662e−004
M-PSO–DE	3.4501e−008	3.4622e−008	1.4035e−004	1.4101e−004
M-PSO–DE–CS	3.0016e−008	3.0498e−008	1.3432e−004	1.3156e−004

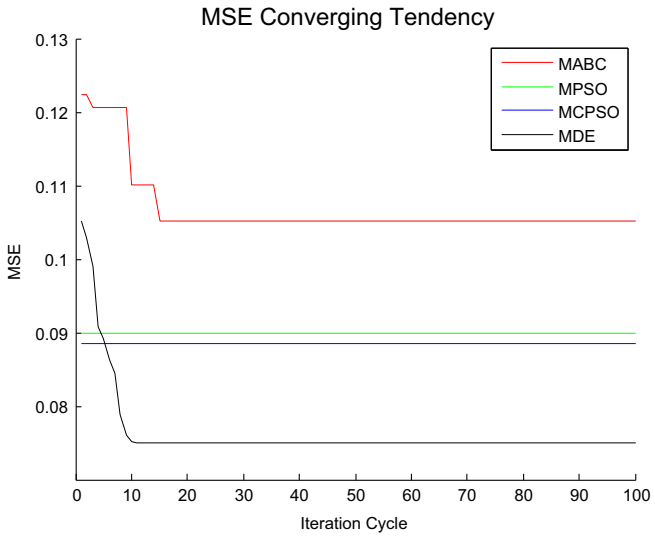


Fig. 5. MSE converging tendency of multi-population I algorithm.

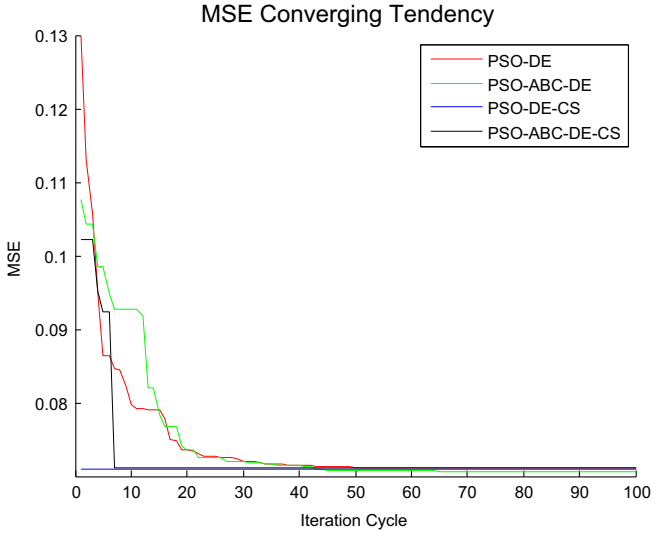


Fig. 6. MSE converging tendency of multi-population II algorithm.

and outperform other single-algorithms or multi-population co-evolution algorithms.

Figs. 4–6 have showed the MSE converging tendency of single-algorithms and multi-population algorithms. Form Fig. 4, we could see that the PSO and CPSO may be easily trapped in the sup-optimum and could not move forward to explore, but ABC and DE could exploit deeply with the iterations. DE has outperformed the other single-algorithms, especially it has more superior converging velocity and precision than ABC and CPSO. From the comparison of Figs. 4 and 5, we could see that the multiple-population algorithms would be more superior than their respective single-algorithms, especially the comparison of ABC and MABC, DE and MDE. PSO with chaos mapping could be improved in some degree, but it does not always work prominently. In addition, we could see that the PSO, CPSO, MPSO, MCPSO could acquire the constant MSE but may be not the optimum. Compared with Figs. 4–6 shows that multiple heterogeneous swarm intelligence algorithm could have superior converging velocity but the approximate precision. Last but not the least, we could draw a conclusion that multiple heterogeneous swarm intelligence could be more excellent than the single-algorithms or multiple homogeneous swarm intelligence (MABC, MPSO, MDE, etc.).

5. Conclusion

Unlike most previous studies just on training the component FNN or the ensemble mechanism, in our paper, the improved model has not only optimized the component FNN, but also meliorated the ensemble mechanism. The construction of our improved NNE model is made up with two phases. The first is adopting CPSO–GD couple algorithm to train each component FNN. The second is optimizing the ensemble mechanism by multi-population co-evolution swarm intelligence.

To demonstrate the superiority of co-evolution algorithm based on multiple heterogeneous swarm intelligence, we have presented the experiments to compare the optimization performance over benchmark functions. In addition, we select 200 samples for training and testing, and make 3 compared tables in which different training algorithms and different ensemble ways have been listed. The superiority of individual networks optimized by different algorithm is analyzed, which reveals that in some cases the ensemble mechanism is superior to the simplex neural network.

Experimental results show that Our M-PSO–ABC–DE–CS and M-PSO–DE–CS algorithm could converge to the global solution and outperform other single-algorithms or multi-population co-evolution algorithms.

Next step, we would research into how to reduce the operating time and calculation scale, at the same time to improve the computational efficiency.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant nos. 60805028, 61175076, 61225017), Natural Science Foundation of Shandong Province (ZR2010FM027), China Postdoctoral Science Foundation (2012M521336), Open Research Project under Grant 20120105 from SKLMCCS, SDUST Research Fund (2010KYTD101), SDUST Graduate Innovation Foundation (YC130218, YC140332).

References

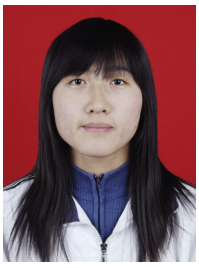
[1] S. Soares, C. Antunes, R. Araújo. A genetic algorithm for designing neural network ensembles, in: ACM Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference, (2012) 681-688.
[2] X. Yao, M.M. Islam, Evolving artificial neural network ensembles, IEEE Comput. Intell. Mag. 3 (1) (2008) 31–42.
[3] Zeng-Shun Zhao, Li Zhang, Meng Zhao, Zeng-Guang Hou, Chang-Shui Zhang, Gabor face recognition by multi-channel classifier fusion of supervised kernel manifold learning, Neurocomputing 97 (2012) 398–404.
[4] L. Cheng, Z.-G. Hou, Y. Lin, M. Tan, W.J. Zhang, F.-X. Wu, Recurrent neural network for non-smooth convex optimization problems with application to the identification of genetic regulatory networks, IEEE Trans. Neural Networks 21 (5) (2011) 714–726.
[5] L. Cheng, Z.-G. Hou, M. Tan, Y. Lin, W.J. Zhang, Neural-network-based adaptive leader-following control for multi-agent systems with uncertainties, IEEE Trans. Neural Networks 21 (8) (2010) 1351–1358.
[6] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990) 993–1001.
[7] M.P. Perrone, L.N. Cooper, When networks disagree: ensemble methods for hybrid neural networks, Neural Networks Speech Image Proc. (1993) 126–142.
[8] H. Chen, X. Yao, Regularized negative correlation learning for neural network ensembles, IEEE Trans. Neural Networks (2009) 1962–1979.
[9] Zhi-Hua Zhou, Jianxin Wu, Wei Tang, Ensembling neural networks: many could be better than all, Artif. Intell. 137 (1-2) (2002) 239–263.
[10] J. Kennedy, R. Eberhart: A discrete binary version of the particle swarm optimization. in: Proceedings IEEE International Conference on Computational Cybernetics and Simulation, Piscatawat. (1997) 4104–4108.

[11] F.L. Minku, T.B. Ludermir, Clustering and co-evolution to construct neural network ensembles: an experimental study, Neural Networks 21 (9) (2008) 1363–1379.
[12] S. Soares, C.H. Antunes, R. Araújo, Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development, Neurocomputing 121 (2013) 498–511.
[13] Marco Gori, Alberto Tesi, On the problem of local minima in back-propagation, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1) (1992) 76–86.
[14] Riccardo Poli, James Kennedy, Tim Blackwell, Particle swarm optimization: an overview, Swarm Intell. 1 (2007) 33–57.
[15] Zengshun Zhao, Jizhen Wang, Qingji Tian, Maoyong Cao: Particle swarm-differential evolution cooperative optimized particle filter. In: International Conference on Intelligent Control and Information Processing. (2010) 485–490.
[16] Swagatam Das, Ajith Abraham, Amit Konar, Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives, Stud Comput. Intell. (SCI) 116 (2008) 1–38.
[17] Bipul Luitel, GaneshK. Venayagamoorthy, Differential evolution particle swarm optimization for digital filter design, in: Proc. 2008 IEEE Congress on Evolutionary Computation (CEC 2008) (2008) 3954–3961 (Crystal city, Washington, DC, USA).
[18] JingRu Zhang, Jun Zhang, TatMing Lok, MichaelR. Lyu, A hybird particle swarm optimization-back-propagation algorithm for feedward neural netwrok train, Appl. Math. Comput. 185 (2007) 1026–1037.
[19] Y. Song, Z. Chen, Z. Yuan, New chaotic PSO-based neural network predictive control for nonlinear process, IEEE Trans. Neural Networks 18 (2) (2007) 595–601.
[20] Bo Liu, Ling Wang, Yi-Hui Jin, Fang Tang, De-Xian Huang, Improved particle swarm optimization combined with chaos, Chaos, Solitons Fractals 25 (2005) 1261–1271.
[21] Hui Yuan, Jun Zhi, Jianyong Liu, Application of particle swarm optimization algorithm-based fuzzy BP neural network for target damage assessment, Sci. Res. Essays 6 (15) (2011) 3109–3121.
[22] P.J. Werbos, Beyond Regression: New Tools for Predictions and Analysis in the Behavioral Science, Ph.D. Thesis, Harvard University. (1974).
[23] T.P. Vogl, J.K Mangis, A.K Rigler, W.T. Zink, D.L. Alkon, Accelerating the convergence of the back-propagation method, Biol. Cybern. 59 (1988) 257–263.
[24] J. Kennedy, R. Eberhart, Particle swarm optimization, In: Proc. IEEE Conf. Neural Networks, Piscataway (1995) 1942–1948.
[25] Zeng-Shun Zhao, Xiang Feng, Yan-yan Lin, Mao-Yong Cao, Zeng-Guang Hou, Min Tan, Improved Rao-Blackwellized particle filter by particle swarm optimization, J. Appl. Math. 2013 (2013) 1–7 (Article ID 302170).
[26] D. Optiz, J. Shavlik, Actively searching for an effectively neural network ensemble, Connection Sci. 8 (3,4) (1996) 337–353.
[27] Giorgio Valentini, Francesco Masulli, Ensembles of Learning Machines. WIRN VIETRI, LNCS (2002) 3–20.
[28] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[29] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (2008) 687–697.
[30] Dervis Karaboga, Bahriye Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC), J. Global Optim. 39 (2007) 459–471.
[31] MohdAfizi Mohd Shukran, YukYing Chung, Wei-Chang Yeh, Noorhaniza Wahid, AhmadMujahid Ahmad Zaida, Artificial bee colony based data mining algorithms for classification tasks, Mod. Appl. Sci. 5 (4) (2011) 217–231.

Zeng-Shun Zhao received the Ph.D. degree in control engineering from the Institute of Automation, Chinese Academy of Sciences, in 2007. He is currently an associate professor at the College of Electrics Communication and Physics, Shandong University of Science and Technology, Qingdao, China. In 2011, he worked as a visiting scientist with Prof. C.S. Zhang. at Stinghua University. His research interests include Machine learning, Pattern Recognition, Computer vision and intelligent robot.

Xiang Feng is master student in Shandong University of Science and Technology. His research interests are computataional intelligence and pattern recognition.

Yan-yan Lin is master student in Shandong University of Science and Technology. Her research interests are computational intelligence and pattern recognition.



Fang Wei is master student in Shandong University of Science and Technology. Her research interests are Image processing and pattern recognition.

Shi-Ku Wang is master student in Shandong University of Science and Technology. His research interests are Image processing and machine learning.

Mao-Yong Cao received the Ph.D. degree in optical engineering from TianJin University China, in 2002. He is a Full Professor at the Shandong University of Science and Technology, Qingdao, China. His research interests include image processing and pattern recognition.

Zeng-Guang Hou received the Ph.D. degree in electrical engineering from Beijing Institute of Technology, Beijing, China, in 1997. He is a Full Professor at the Institute of Automation, Chinese Academy of Sciences, Beijing, China. Professor Hou is currently serving as an Associate Editor for IEEE Transaction on Systems, Man, and Cybernetics and IEEE Transactions on Neural Networks. His research interests include neural networks, optimization algorithms, robotics, and intelligent control systems.