Dendritic Neuron Model With Effective Learning Algorithms for Classification, Approximation, and Prediction

Shangce Gao, *Senior Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, Yirui Wang, Jiujun Cheng, Hanaki Yachi, and Jiahai Wang, *Member, IEEE*

***Abstract*— An artificial neural network (ANN) that mimics the information processing mechanisms and procedures of neurons in human brains has achieved a great success in many fields, e.g., classification, prediction, and control. However, traditional ANNs suffer from many problems, such as the hard understanding problem, the slow and difficult training problems, and the difficulty to scale them up. These problems motivate us to develop a new dendritic neuron model (DNM) by considering the nonlinearity of synapses, not only for a better understanding of a biological neuronal system, but also for providing a more useful method for solving practical problems. To achieve its better performance for solving problems, six learning algorithms including biogeography-based optimization, particle swarm optimization, genetic algorithm, ant colony optimization, evolutionary strategy, and population-based incremental learning are for the first time used to train it. The best combination of its user-defined parameters has been systemically investigated by using the Taguchi's experimental design method. The experiments on 14 different problems involving classification, approximation, and prediction are conducted by using a multilayer perceptron and the proposed DNM. The results suggest that the proposed learning algorithms are effective and promising for training DNM and thus make DNM more powerful in solving classification, approximation, and prediction problems.**

***Index Terms*— Approximation, brain, classification, dendritic neuron model (DNM), global learning algorithms, prediction.**

I. Introduction

THE brain comprised of 10^{11} neurons with generally 10^{15} interconnections among them can handle massive information. Their shape and size are diverse and their structures are distinct. The components of a neuron can be categorized as cell body, dendrite, and axon. The received signal from other neurons is reckoned at the synapse and sent to the cell body. Once the signal entering the cell body surpasses the sustaining threshold, the neuron is burned and the signal is sent to other neurons by the axon. McCulloch and Pitts [1] proposed a plain neuron model where there was no influence between dendrites and synapses, their functions were regarded as the weight and transmission, and the cell body was deemed to be the primary calculation unit. Minsky and Papert [3] presented that some basic calculation could not be implemented via one-layer McCulloch–Pitts cells. Although a simple two-layer model is deficient in resolving some approximating functions or other applications, it offers a prospective way to develop a multilayer network for enhancing its performance. Then, multilayer feedforward networks are investigated and explored to further refine the property of the model consisting of neurons [2].

Artificial neural networks (ANNs) [3], [4] adopt more than one neuron to handle the received information, and are effective methods for solving many optimization problems, such as the cyclic motion of humanoid robots implemented by a neural-dynamic method using a recurrent neural network [5] and repetitive motion of redundant robot manipulators achieved by three networks (dual one, linear variational inequality (LVI)-based primal-dual one, and simplified LVI-based primal-dual one) [6]. Yet some attentions have been attracted on single neuron models, such as a single multiplicative neuron model [7]–[9], branch-specific plastic neuron [10]–[12], and sigma–pi unit [13]–[19]. Nevertheless, these models depend on the structure of a McCulloch–Pitts neuron, which utilizes weights between synapses to describe their clustering extent. Hence, all the information of locality in dendrites is missing, and these models can hardly reflect local interactions within a stationary dendritic tree. Furthermore, these McCulloch–Pitts-based single neuron models, e.g., multiplicative neuron [7] and sigma–pi neuron [17], are restricted when addressing complicated problems owing to their nonlinear calculative abilities [20], especially the nonlinearly separated problems [21].

Recently, several theoretical studies [22]–[24] have inferred that a single neuron retreated as a calculative unit could perform more effectively by taking into account the synaptic nonlinearities in a dendritic tree, which is significantly different from the McCulloch–Pitts neuron-based models. The diverse kinds of synaptic plasticity and nonlinearity mechanisms enable synapses to take a valuable part of calculation [25].

Manuscript received November 8, 2017; revised March 3, 2018; accepted June 7, 2018. Date of publication July 10, 2018; date of current version January 21, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61673403, Grant U1611262, and Grant 61472284, in part by the Opening Project of Guangdong High-Performance Computing Society under Grant 2017060109, and in part by JSPS KAKENHI under Grant JP17K12751. *(Corresponding authors: MengChu Zhou; Jiujun Cheng.)*

S. Gao, Y. Wang, and H. Yachi are with the Faculty of Engineering, University of Toyama, Toyama 930-8555, Japan (e-mail: gaosc@eng.u-toyama.ac.jp; wyr607@foxmail.com; hanaki.fatidll@gmail.com).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: mengchu.zhou@njit.edu).

J. Cheng is with the Department of Computer Science and Technology, Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China (e-mail: chengjj@tongji.edu.cn).

J. Wang is with the Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, China (e-mail: wangjiah@mail.sysu.edu.cn).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

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.2018.2846646

Synaptic inputs from various neurons can be distributed spatially in a dendritic tree. Changes within synaptic strength or connectivity and the excitability of neurons can result in their plasticity [26]. Besides, a slightly morphological difference can lead to the notable functional discrepancy, playing a role in filters to determine the types of signals that single neurons receive and the way used to integrate the signals [27].

By taking into consideration the nonlinearity of synapses, a single dendritic neuron model (DNM) has been proposed [28]–[30] and successfully applied to breast cancer classification [29], liver disorders diagnosis [31], financial time series prediction [32], tourist arrivals forecasting [33], and morphological hardware realization [30]. According to [33], the characteristics of DNM can be summarized as follows: 1) its architecture is similar to those of multiplicative neuron models and sigma–pi models. They are multilayered, and signals are transferred in a feedforward manner. Thus, the functions used in these models can be reciprocated. For example, the radial basis functions using Gaussian kernels, a simplified fuzzy logic formulation, and kernel-regression models are able to be represented by a variation of the sigma–pi formulation [34]. Furthermore, some of them are isomorphic (e.g., the augmented two-layer neuron model 2LM is isomorphic to a traditional ANN [35]); and 2) multiplication is both the simplest and one of the most widespread of all nonlinear operations in the nervous system [36]. Taking the advantage of the multiplication operation that is essential and important to the information processing in a neuron [20], the computation in synapses is innovatively modeled by using sigmoid functions. Depending on the values of the parameters in synapses, the output of synapses can successfully represent excitatory, inhibitory, and constants 0 and 1 signals, which is beneficial for identifying the morphology of a neuron [28].

Although DNM has achieved great successes in various applications, all of them utilize a back-propagation (BP) learning algorithm to train DNM. In BP learning [37] that is essentially a gradient descent optimization method [38], derivatives of the node functions are required and the minimization of a nonconvex error function over high-dimensional spaces is implemented. It is well-known that BP and its improved versions (e.g., Rprop [39], [40] and Quickprop [41]) suffer from the difficulties originated from the proliferation of saddle points and local minima [42]. Much work has been done by adopting non-BP learning algorithms [43]–[53], i.e., those algorithms that require no derivative calculation needed but use the improvement of a solution based on comparison, to perform neural network learning. This paper aims to identify effective learning algorithms to train DNM and explore its applications.

In this paper, six non-BP learning algorithms, i.e., biogeography-based optimization (BBO), particle swarm optimization (PSO), genetic algorithm (GA), ant colony optimization (ACO), evolutionary strategy (ES), and population-based incremental learning (PBIL) are used to train DNM. These algorithms are the primary and popular intelligent optimization approaches in the field of evolutionary computation. Moreover, they have been verified to be effective and promising for training a multilayer perceptron (MLP) [54]. Therefore, we select them to further enhance the performance of DNM. It is worth pointing out that differential evolution that has good performances for benchmark optimization functions is not adopted in this paper to train DNM because it generally undergoes a stagnation and fails to enhance its performance during neural network training [55]. According to the no free lunch theorem [56], there is no such algorithm that can suit to solving all optimization or learning problems. Thus, these six widely used algorithms are all tested to find out: 1) which learning algorithm is the most suitable for DNM and 2) how promising a well-learned DNM can effectively solve classification, approximation, and prediction problems. Extensive application instances involving the N-bit XOR classification, balloon classification, Iris classification, breast cancer classification, heart classification, sigmoid function approximation, cosine function approximation, sine function approximation, sphere function approximation, Griewank function approximation, Rosenbrock function approximation, Mackey–Glass prediction, Box–Jenkins chaotic data prediction, and electroencephalography (EEG) data prediction are adopted as a test suite to verify the performance of DNM. As a preliminary experiment, the best combination of user-defined parameters has been systemically investigated by using the Taguchi's experimental design method [57]. Then, all algorithms are implemented for the selected 14 different problems on both MLP and DNM, respectively. The results suggest that the proposed learning algorithms are effective and promising for training DNM and using it in solving classification, approximation, and prediction problems.

This work contributes to communities of ANNs and evolutionary computation in the following aspects: 1) a novel DNM [28]–[33] by considering the nonlinearity of synapses is developed. For a given task, the proposed DNM is able to reconstruct the morphological structure of a neuron, not only enabling us to achieve the deep understanding of a single neuron computational mechanism, but also providing an effective and efficient problem solver from an engineering perspective; 2) its extensive applications involving classification, approximation, and prediction are shown. Thus, its wide industrial applications can be expected, such as pattern recognition problem, logic circuit layout, product sales forecasting, and job shop scheduling; 3) similar to MLP that has been widely applied in many fields, such as engineering, bioinformation, and industry [58]–[61], DNM is established to become an important method as an alternative in these fields; and 4) user-defined parameters have been systemically investigated to obtain a reasonable (if not the best) combination of parameter settings for six learning algorithms. Therefore, it provides some insights into them and can help us better understand the performance of DNM.

The rest of this paper is organized as follows. Section II introduces the fundamental structures and functions of DNM. Section III presents six learning algorithms for training DNM in detail. Section IV shows the sensitivity analysis of user-defined parameters and experimental results of MLP and DNM with various learning algorithms on 14 different problems. Section V draws important conclusions.

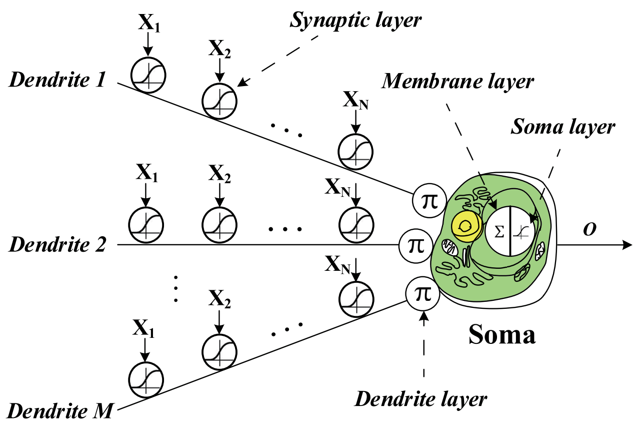


Fig. 1. Structure of DNM.

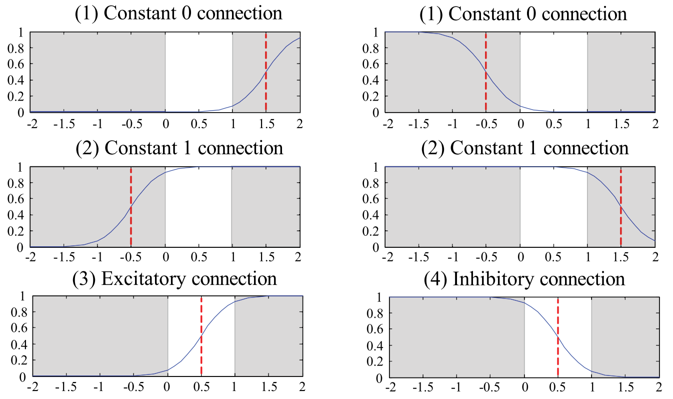


Fig. 2. Four types of connection instances in the synaptic layer.

II. Dendritic Neuron Model

DNM is composed of four layers. Its synaptic layer possesses a sigmoid function for received inputs. Its dendrite layer has a multiplicative function for the outputs of its synaptic layer. Its membrane layer performs an additional function for the outputs of its dendrite layer. Its soma layer utilizes another sigmoid function for the outputs of its membrane layer and gives its final results. Its details are described as follows and its structure is shown in Fig. 1.

A. Synaptic Layer

A synapse connects neurons from a dendrite to another dendrite/axon or the soma of another neural cell. The information flows from a presynaptic neuron to a postsynaptic neuron, which shows feedforward. The changes in the postsynaptic potential influenced by ionotropic determine the excitatory or inhibitory of a synapse. The description connecting the ith ($i = 1, 2, \ldots, N$) synaptic input to the jth ($j = 1, 2, \ldots, M$) synaptic layer is given as

$$Y_{ij} = \frac{1}{1 + e^{-k(w_{ij}x_i - \theta_{ij})}} \tag{1}$$

where Y_{ij} is the output from the ith synaptic input to the jth synaptic layer. k indicates a positive constant. x_i manifests the ith input of a synapse and $x_i \in [0, 1]$. Weight w_{ij} and threshold θ_{ij} are the connection parameters to be learned.

According to the values of w_{ij} and θ_{ij}, four types of connection instances are shown in Fig. 2, where the horizontal axis indicates the inputs of presynaptic neurons and the vertical one clarifies the output of the synaptic layer. As the range of x is [0, 1], only the conforming part is required to be seen. The four connection instances contain: 1) a constant 0 connection (when $w_{ij} < 0 < \theta_{ij}$ or $0 < w_{ij} < \theta_{ij}$) where the output is approximately 0 no matter when the input transforms from 0 to 1; 2) a constant 1 connection (when $\theta_{ij} < w_{ij} < 0$ or $\theta_{ij} < 0 < w_{ij}$) where the output is approximately 1 no matter when the input transforms from 0 to 1; 3) the excitatory connection (when $0 < \theta_{ij} < w_{ij}$) where the output is proportional to the input no matter when the input transforms from 0 to 1; and 4) the inhibitory connection (when $w_{ij} < \theta_{ij} < 0$) where the output is inversely proportional to the input no matter when the input transforms from 0 to 1. It is worth to note that these four connection instances are critical to infer the morphology of a neuron by specifying the positions and synapse types of dendrites.

B. Dendrite Layer

The dendrite layer shows a multiplicative function of the outputs from synapses at various synaptic layers. A type of multiplicative operations can be achieved due to the nonlinearity of synapses, i.e., constant 0 or 1 connection. That is why a multiplicative operation has been chosen to use in this model when it comes to the dendrite layer. The multiplication is equivalent to the logic AND operation since the values of inputs and outputs of the dendrites correspond to 1 or 0. In Fig. 1, symbol π stands for a multiplicative operator. The output function for the jth dendrite branch is expressed as follows:

$$Z_j = \prod_{i=1}^{N} Y_{ij}. \tag{2}$$

C. Membrane Layer

The membrane layer collects the signals from each dendritic branch. The input received from a dendrite branch is calculated with a summation function, which closely resembles a logic OR operation. Then, the resultant output is delivered into the next layer to activate the soma body. The output of this layer is formulated as

$$V = \sum_{j=1}^{M} Z_j. \tag{3}$$

D. Soma Layer

At last, the soma layer implements the function of soma body such that the neuron fires if the output from the membrane layer exceeds its threshold. This process is expressed by a sigmoid function used to calculate the ultimate output of the entire model

$$O = \frac{1}{1 + e^{-k_s(V - \theta_s)}}. \tag{4}$$

The parameter k_s is a positive constant and the threshold θ_s alters from 0 to 1.

Fig. 1 describes a complete structure of DNM. Inputs $X_1, X_2, \ldots,$ and X_N in each dendrite are firstly transformed to their corresponding outputs according to four connection instances in the synaptic layer. Second, all the outputs from the synaptic layer in each dendrite are multiplied as new outputs of the dendrite layer. Third, all the outputs in the dendrite layer are summed to obtain an output of the membrane layer. Finally, this output of the membrane layer is regarded as the input of the soma layer to calculate the ultimate result of DNM.

III. Learning Algorithms for Dendritic Computation

Conventional DNM uses BP to adjust the weights and thresholds in it [28]–[30]. However, BP is a limited learning algorithm due to its inherent local-minimum trapping problem and usually cannot find the global best values of its weights and thresholds. Thus, its performance is limited. To achieve its better performance for resolving complex application problems, six learning algorithms are used to train it in this paper.

A. Biogeography-Based Optimization

Inspired by the biogeography, BBO was proposed by Simon [62]. Biogeography is the investigation of the speciation, extinction, and geographical distribution of biological species. Mathematical models of biogeography exhibit speciation, the migration of species among islands, and the extinction of species. Geographical areas that are kind to life represent suitable habitats defined by the habitat suitability index (HSI). Attributes related with HSI are called the suitability index variables (SIVs) [63].

From the optimization perspective, habitats are treated as search agents, which are analogous to the chromosomes in GAs. A vector of habitants is assigned to each habitat in BBO. The vector represents the search variables in an optimization problem. HSI is used to measure the fitness (or affinity) of a habitat. Without loss of generality, for a minimization problem, the lower HSI is, the better the habitat is. For an optimization problem, BBO evolves its habitats based on three basic rules: 1) habitants in the habitat with a better HSI (i.e., fitness) are more likely to emigrate its SIV (i.e., component of variables) to the habitat with worse HSI; 2) those habitats with worse HSI tend to attract new immigrant habitants from those with better HSI; and 3) habitats also have a chance to randomly change their habitants in spite of their HSI.

As a result, the immigration rate λ_i for each habitat is a monotonically nonincreasing function of HSI and the emigration rate μ_i is a monotonically nondecreasing function of HSI, which are defined as follows:

$$\lambda_i = I \times \left(1 - \frac{n}{Q}\right) \tag{5}$$

$$\mu_i = E \times \frac{n}{Q} \tag{6}$$

where I and E are the maximum immigration rate and the emigration rate, respectively. n ($n = 0, 1, 2, \ldots, Q$) is the current rank of habitat. The better the habitat, the higher its rank.

Algorithm 1: Habitat Migration

```
begin
    for $i$=1 to $Q$ do
        Choose $Habitat_i$ with probability $\propto \lambda_i$;
        if $rnd\ (0,1) < \lambda_i$ then
            for $j$=1 to $Q$ do
                Choose $Habitat_j$ with probability $\propto \mu_j$
                if $rnd\ (0,1) < \mu_j$ then
                    Randomly choose a SIV $\delta$ from $Habitat_j$

                    Substitute a random SIV in $Habitat_i$
                     with $\delta$
```

The mutation in BBO improves its exploration ability and keeps habitats as diverse as possible, defined as follows:

$$m_s = m_{\max} \times \left(\frac{1 - p_s}{p_{\max}}\right) \tag{7}$$

where $m_{\max}$ is an initial value of mutation. p_s is the mutation probability of the nth habitat, and $p_{\max} = \text{argmax}(p_s)$, $s = 1, 2, \ldots, Q$.

To train DNM, BBO is implemented as follows.

1) Initialize BBO parameters. In DNM training, the habitats indicate solutions of the DNM weights and thresholds in synaptic connections. Formally, they can be expressed as

$$\begin{aligned}\text{Habitat}_i &= \{\text{SIV}_1, \text{SIV}_2, \ldots, \text{SIV}_m\} \\ &= \{w_{11}, w_{12}, \ldots, w_{\text{MN}}, \theta_{11}, \theta_{12}, \ldots, \theta_{\text{MN}}\}\end{aligned} \tag{8}$$

where Habitat_i ($i = 1, 2., \ldots, Q$) denotes a solution in BBO, and Q is the population size. It is clear that $m = 2\text{MN}$. The maximum immigration and the maximum migration rates are set as $I = E = 1$, the maximum mutation rate is set as $p_{\max} = \text{argmax}(p_s)$, and an elitism parameter is adopted, i.e., the two best habitats are always kept from one generation to the next.

2) Initialize a random set of habitats, and each habitat corresponds to a solution.

3) Calculate the HSI of each habitat by using the following equation:

$$\text{HSI}(\text{Habitat}_i) = \text{MSE}(\text{Habitat}_i) = \frac{1}{2P}\sum_{p=1}^{P}(T_p - O_p)^2 \tag{9}$$

where P is the total number of training samples, T_p is the target vector of the pth sample, and O_p is the actual output vector obtained by Habitat_i. Besides, the immigration and emigration rates are calculated via (5) and (6).

4) Probabilistically utilizes the operation of immigration and emigration to improve each non-elite habitat according to Algorithm 1.

Algorithm 2: Habitat Mutation

```
begin
    for $s$=1 to $Q$ do
        Calculate the probability $p_s$
        Choose SIV $Habitat_s(j)$ with probability $\propto p_s$
        if $rnd\ (0,1) < m_s$ then
            Substitute $Habitat_s(j)$ with a randomly formed SIV
```

5) Update the probability m_s of each habitat via (7). Thereafter, the mutation of each non-elite habitat is implemented according to Algorithm 2. *rnd* (0, 1) indicates a uniformly distributed random real number in the interval (0, 1) and $\text{Habitat}_s(j)$ denotes the jth SIV of solution Habitat_s.

6) Go to Step 3 for the next iteration. The algorithm does not end until the termination condition is satisfied.

B. Particle Swarm Optimization

PSO proposed by Eberhart *et al.* [64] is to mimic the search behavior of a bird swarm. The PSO algorithm consists of particles, and each particle embodies a possible solution. Each particle has two attributes, i.e., the position and velocity of one particle. Particles arrive in new positions via moving in the search space. After some iterations, particles can move toward the optimal solution according to two fundamental principles: a cognitive principle and a social principle. The first one is to find the best position of the particle itself (pbest). The second one is to use the best position in the whole particles (gbest) to direct them to go toward a better place. Each particle updates its own velocity that guides it to the next position by following these two principles.

PSO has been usually used to train neural networks [45], [65]–[68], especially MLP [54]. In this paper, we use PSO as an optimizer to seek for the best combination of the values of weights and thresholds in synaptic connections, that is,

$$\begin{aligned} X_i &= \left\{x_i^1, x_i^2, \ldots, x_i^m\right\} \\ &= \{w_{11}, w_{12}, \ldots, w_{\text{MN}}, \theta_{11}, \theta_{12}, \ldots, \theta_{\text{MN}}\} \end{aligned} \tag{10}$$

where X_i $(i = 1, 2, \ldots, Q)$ denotes the ith individual in the population, and Q is the population size. PSO is used to train DNM using the procedures described in Algorithm 3, where $\text{MSE}(X_i)$ calculates the output error of DNM for a given solution X_i and it is defined as follows:

$$\text{MSE}(X_i) = \frac{1}{2P}\sum_{p=1}^{P}(T_p - O_p)^2. \tag{11}$$

Algorithm 3: PSO

```
begin
    Initialize a random population of individuals $\{X_i\}$, $i \in [1, Q]$
    Initialize each individual's $m$-element velocity vector $V_i$, $i \in [1, Q]$
    Initialize the best-so-far position of each individual: $B_i \leftarrow X_i$, $i \in [1, Q]$
    Define the neighborhood size $\Delta < Q$
    Define the maximum influence values $\phi_{1,max}$ and $\phi_{2,max}$
    Define the maximum velocity $V_{max}$
    while Termination criterion do
        for each individual $X_i$ do
            $H_i \leftarrow \{\Delta$ nearest neighbors of $X_i\}$
            $P_i \leftarrow \text{argmin}_X\{\text{MSE}(X) : X \in H_i\}$
            Generate a random vector $\phi_1$ with $\phi_1(k) \sim U[0, \phi_{1,max}]$ for $k \in [1, m]$
            Generate a random vector $\phi_2$ with $\phi_2(k) \sim U[0, \phi_{2,max}]$ for $k \in [1, m]$
            $V_i \leftarrow V_i + \phi_1(B_i - X_i) + \phi_2(P_i - X_i)$
            for $|V_i| > V_{max}$ do
                $V_i \leftarrow \frac{V_i V_{max}}{|V_i|}$
            $X_i \leftarrow X_i + V_i$
            $B_i \leftarrow argmin\{\text{MSE}(f(X_i), \text{MSE}(B_i))\}$
```

C. Genetic Algorithm

GAs are simulations of natural selection. According to [63], given an optimization problem, we create a population of candidate solutions called individuals. The good individuals have a relatively high chance of reproducing, while the poor ones have a relatively low chance of reproducing. Parents beget children, and then they drop out of the population to make ways for their offspring. As generations come and go, the population becomes more fit. Sometimes one or more "supermen" evolve to become highly fit individuals that can provide near-optimal solutions to our engineering problem. We can also use GA to train DNM.

Similar to PSO, a chromosome (i.e., individual) in GA for training DNM can be represented using (10), where X_i $(i = 1, 2, \ldots, Q)$ denotes the ith chromosome in the population, and Q is the population size. We use a single-point crossover to update individuals. The elite strategy is also adopted to determine how many best individuals are to be kept from one generation to the next. In addition, the fitness function is the same as (11). GA described in Algorithm 4 is used to train DNM.

D. Ant Colony Optimization

ACO is a metaheuristic algorithm proposed by Dorigo *et al.* [69]. It emulates the foraging behavior of ants to search the shortest route between a food source and their nest. Its key factor is the pheromone deposited by ants, which enables ants to communicate with each other. The amount of pheromone guides an ant to select one path. With time, the amount of pheromone on a path evaporates and the path becomes less attractive in order to let ants explore other paths.

ACO was originally developed for discrete problems such as traveling salesman problems [70]. It has been modified for

Algorithm 4: GA

```
begin
    Determine the initial random population of
     chromosomes ($\{X_i\}$, $i \in [1, Q]$ )
    Evaluate the fitness MSE($X_i$) of the chromosomes
     using Eq. (11)
    while Termination criterion do
        Choose the best chromosomes to be manipulated in
         the current population using Roulette Wheel
         Selection;
        Generate new chromosomes using one-point
         crossover and mutation;
        Evaluate the fitness of the new chromosomes;
        Use the best new chromosomes to replace the
         worst chromosomes of the population;
```

Algorithm 5: ACO

```
begin
    $m$: number of dimensions of the optimization
     problem;
    Divide the $j$th dimension into $B_j$-1 intervals as shown
     in Eq. (13), $j \in [1, m]$;
    $\alpha$: importance of pheromone amounts;
    $S$: deposition constant;
    $\rho$: evaporation rate $\in (0, 1)$;
    $\tau^{jk} = \tau_0$ (initial pheromone) for $j \in [1, m]$ and
     $k \in [1, B_j - 1]$;
    Randomly initialize a population of ants (candidate
     solutions) $X_i$, $i \in [1, Q]$;
    while Termination criterion do
        for each ant $X_i$, $i \in [1, Q]$ do
            for each dimension $j \in [1, m]$ do
                for each discretized interval $[b^{jk}, b^{j,k+1}]$,
                 $k \in [1, B_j - 1]$ do
                    Probability $p^i_{ij} = \frac{\tau^{\alpha}_{jk}}{\sum_{l=1}^{B_j - 1} \tau^{\alpha}_{jl}}$;
                $X_i(x^j) = U[b^{jk}, b^{j,k+1}]$ with probability
                 $p^i_{ij}$;
        $L_i \leftarrow$ cost of solution constructed by ant $X_i$,
         $i \in [1, Q]$;
        for each dimension $j \in [1, m]$ do
            for each discretized interval $[b^{jk}, b^{j,k+1}]$,
             $k \in [1, B_j - 1]$ do
                for each ant $X_i$, $i \in [1, Q]$ do
                    if $X_i(x_j) \in [b^{jk}, b^{j,k+1}]$ then
                        $\Delta\tau^i_{jk} = \frac{S}{L_i}$;
                    else
                        $\Delta\tau^i_{jk} = 0$;
                $\tau_{jk} = (1 - \rho)\tau_{jk} + \sum_{i=1} Q\Delta\tau^i_{jk}$;
```

optimization problems in continuous domains [71]. To realize it, each dimension j of the search space can be divided into discretized intervals. In other words, the m-dimensional optimization function $f(X_i)$, where $X = [x_i^1, x_i^2, \ldots, x_i^n]$, is to be minimized, and

$$x_i^j \in \left[x_i^{j,\min}, x_i^{j,\max}\right] \tag{12}$$

$$x_i^{j,\min} = b^{j1} < b^{j2} < \cdots < b^{j,B_j} = x_i^{j,\max} \tag{13}$$

where $B_j - 1$ is the number of discrete intervals into which we divide the jth domain. In each generation, if the jth domain of a solution is between b^{jk} and $b^{j,k+1}$, then we update the pheromone of that interval as that in the traditional ACO [69], [71], i.e.,

$$\tau^{jk} = \tau^{jk} + \frac{S}{f(X_i)} \quad \text{if } x_i^j \in [b^{jk}, b^{j,k+1}] \tag{14}$$

where S is a deposition constant that is defined by users and $f(X_i) = \text{MSE}(X_i) = 1/(2P)\sum_{p=1}^{P}(T_p - O_p)^2$ for the training of DNM. We use pheromone amounts to probabilistically construct new solutions at the beginning of each generation. If the interval $[b^{jk}, b^{j,k+1}]$ has a lot of pheromone, then there is a large probability that a candidate solution is constructed such that its jth dimension is in that interval. A simple method of realizing this is to set the jth dimension of the solution to a random number that is uniformly generated from $[b^{jk}, b^{j,k+1}]$.

Similar to the above-mentioned optimizers, an ant in ACO is represented via (10). ACO is implemented in Algorithm 5.

E. Evolutionary Strategy

Evolutionary strategies [72], [73] are inspired by the concept of the evolution. ESs contain both genotypic and phenotypic evolutions. Each individual is constructed by its genetic building blocks and several strategy parameters. Both genetic characteristics and strategy parameters evolve, while the latter determine the evolution of the former.

ES is composed of five basic components.

1) *Initialization:* Each individual's genotype is set to be within the problem boundary constraints, and the strategy parameters are also set.
2) *Recombination:* A crossover operator on two or more parents generates offspring.
3) *Mutation:* Self-adaptive strategy parameters change the component of offspring.
4) *Evaluation:* The quality of an individual is assessed via the fitness function.
5) *Selection:* Selection operators are executed to choose parents for recombination and confirm the individuals in the next generation.

ES is implemented in Algorithm 6.

F. Population-Based Incremental Learning

An estimation of distribution algorithm (EDA) optimizes a function by keeping track of the statistics of the population of candidate solutions [63], [74]–[76]. Since the statistics of the population are maintained, the actual population itself does not need to be maintained from one generation to the next. A population is created at each generation from the previous

Algorithm 6: ES

```
begin
    $\mu$: the number of parents;
    $\lambda$: the number of offspring;
    Set the generation counter $t = 0$;
    Initialize the strategy parameters;
    Initialize the population $\{X(0)\}$ of $\mu$ individuals;
    for each individual $X_i(t) \in X(t)$ do
        Evaluate the fitness
         $\mathrm{MSE}(X_i) = \frac{1}{2P}\sum_{p=1}^{P}(T_p - O_p)^2$;
    while Termination criterion do
        for $i = 1, \ldots, \lambda$ do
            Randomly choose 1 or 2 parents;
            Use the crossover operator to generate
             offspring;
            Mutate offspring's strategy parameters and
             genotype;
            Evaluate the fitness of offspring;
        Choose the new population $X(t+1)$;
        $t = t + 1$;
```

generation's population statistics, and then the statistics of the fittest individuals in the population are computed. Finally, a new population is generated by using the statistics of the fittest individuals. This process repeats from one generation to the next. EDAs are population-based algorithms that discard at least part of the population in each generation and replace it using newly generated solutions that are sampled based on the statistical properties of highly fit individuals in the current population.

PBIL is a generalization of a univariate marginal distribution algorithm. Given an m-dimensional binary optimization problem, PBIL maintains an m-dimensional probability vector p. The kth element of p specifies the probability that the kth element of a candidate solution is equal to 1. PBIL is motivated by competitive learning, which is a simple method of learning in ANNs. At each generation, we use the probability vector p to probabilistically generate a random set of candidate solutions. Subsequently, we evaluate the fitness of each candidate solution. Next, we adjust the probability vector such that the next generation is more likely to be similar to the fittest individuals, and less likely to be similar to the least fit individuals. Given this new probability vector, we proceed to the next generation by using p to create another random population of candidate solutions. This process continues until the user-defined convergence criterion is satisfied. PBIL for training DNM is implemented in Algorithm 7.

IV. Experiment and Analysis

A. Experimental Setup

In our experiments, five classifications, six function approximations, and three predictions are used to verify the performance of DNM together with the above-mentioned six learning algorithms. The classification data sets, i.e., XOR, balloon, iris, breast cancer, and heart are acquired from the University of California at Irvine Machine Learning Repository [77]. Table I summarizes their number of attributes, number of training samples, number of test samples, and number of classes. The function approximation data sets are 1-D sigmoid, 1-D cosine with one peak, 1-D sine with four peaks, 2-D sphere, 2-D Griewank, and 5-D Rosenbrock functions. Table II lists their function expressions, the number and value range of training samples, and the number and value range of testing samples. Table III represents the details involving the number of training and testing samples for the Mackey Galss, Box Jenkins, and EEG times series data. The Mackey–Glass equation is derived from a nonlinear time delay differential

Algorithm 7: PBIL

```
begin
    A candidate solution is presented by the Eq. (10);
    $Q$: population size;
    $Q_{best}$: number of good individuals that are used to
     adjust $p$;
    $Q_{worst}$: number of bad individuals that are used to
     adjust $p$;
    $p_{max} \in [0, 1]$: maximum allowable value of $p$;
    $p_{min} \in [0, 1]$: minimum allowable value of $p$;
    $\eta \in (0, 1)$: learning rate;
    Initialize the $m$-element probability vector
     $p = [0.5, \ldots, 0.5]$;
    while Termination criterion do
        Use $p$ to randomly generate $Q$ individuals $X_i$ as
         follows:
        for $i \in [1, Q]$ (for each individual) do
            for $k \in [1, m]$ (for each bit) do
                $r \leftarrow$ random number in $U[0, 1]$;
                if $r < p_k$ then
                    $X_i(k) = 0$
                else
                    $X_i(k) = 1$
        Sort the individuals so that
         $\mathrm{MSE}(X_1) \le \mathrm{MSE}(X_2) \le \ldots \le \mathrm{MSE}(X_Q)$, where
         $\mathrm{MSE}(X_i) = \frac{1}{2P}\sum_{p=1}^{P}(T_p - O_p)^2$;
        for $i \in [1, Q_{best}]$ do
            $p = p + \eta(X_i - p)$
        for $i \in [Q - Q_{worst} + 1, Q]$ do
            $p = p - \eta(X_i - p)$
        Probabilistically mutate $p$;
        $p = max(min(p, p_{max}), p_{min})$;
```

TABLE I
Details of the Classification Data Sets

Classification datasets	# of attributes	# of training samples	# of test samples	# of classes
3-bits XOR	3	8	8	2
Balloon	4	16	16	2
Iris	4	150	150	2
Breast cancer	9	599	100	2
Heart	10	297	297	2

TABLE II

DETAILS OF THE FUNCTION APPROXIMATION DATA SETS

Function approximation datasets	# of Training samples	# of Test samples
Sigmoid: $y = 1(1 * e^{-x})$	61: x in [-3:0.1:3]	121: x in [-3:0.05:3]
Cosine: $y = (cos(x\pi/2)^7$	31: x in [1.25:0.05:2.75]	38: x in [1.25:0.04:2.75]
Sine: $y = sin(2x)$	126: x in [-2π:0.1:2π]	252: x in [-2π:0.05:2π]
Sphere: $z = \sum_{i=1}^{2} x_i^2$, $x = x_1$ and $y = x_2$	21×21: x,y in [-2:0.2:2]	41×41: x,y in [-4:0.1:2]
Griewank: $z = \frac{1}{4000}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2} cos(\frac{x_i}{\sqrt{i}}) + 1$, $x = x_1$ and $y = x_2$	21×21: x,y in [-2:0.1:2]	41×41: x,y in [-4:0.05:2]
Rosenbrock: $z = \sum_{i-1}[x_i^2 - 10cos(2\pi x_i) + 10]$, $i = 1, 2, 3, 4, 5$	1024: $x_1 - x_5$ in [-5:3:5]	7776: $x_1 - x_5$ in [-5:2:5]

TABLE III

DETAILS OF THE PREDICTION DATA SETS

Prediction datasets	# of training samples	# of test samples
Mackey Glass	450	550
Box Jenkins	140	156
EEG	1000	1500

TABLE IV

INITIAL PARAMETERS USED IN LEARNING ALGORITHMS

Algorithm	Parameter	Value
BBO	Habitat modification probability	1
	Immigration probability bounds per gene	[0,1]
	Step size for numerical integration of probabilities	1
	Max immigration (I) and Max emigration (E)	1
	Mutation probability	0.005
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of generations	250
PSO	Topology	Fully connected
	Cognitive constant(C_1)	1
	Social constant(C_2)	1
	Inertia constant(w)	0.3
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of iterations	250
GA	Type	Real coded
	Selection	Roulette wheel
	Crossover	Single point (probability = 1)
	Mutation	Uniform (Probability = 0.01)
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of generations	250
ACO	Initial pheromone (τ_0)	1e-0.6
	Pheromone update constant(Q)	20
	Pheromone constant(q_0)	1
	Global pheromone decay rate(p_g)	0.9
	Local pheromone decay rate(p_l)	0.5
	Pheromone sensitivity(α)	1
	Visibility sensitivity(β)	5
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of iteration	250
ES	λ	10
	σ	1
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of generations	250
PBIL	Learning rate	0.05
	Good population member	1
	Bad population member	0
	Elitism parameter	1
	Mutational probability	0.1
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of generations	250
BP	Learning rate	0.01
	Maximum number of generations	3000

TABLE V

REASONABLE COMBINATION OF FOUR DNM PARAMETERS FOR 14 TESTED PROBLEMS, RESPECTIVELY

Problem	M	k	k_s	θ_s
XOR	5	25	25	0.3
Balloon	20	1	25	0.9
Iris	5	25	25	0.3
Breast Cancer	10	1	15	0.7
Heart	10	25	1	0.9
Sigmoid	10	1	15	0.7
Cosine	1	15	25	0.5
Sine	3	5	25	0.7
Sphere	20	5	1	0.3
Griewank	10	10	25	0.1
Rosenbrock	20	5	1	0.3
Mackey Glass	3	1	5	0.3
Box Jenkins	3	1	5	0.3
EEG	5	1	10	0.5

TABLE VI

EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM XOR

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	1.22E-06 (4.31E-22)	8.3	4.57E-07 (1.62E-22)	5.8	0.0	465.0	0.649855
PSO	1.31E+00 (2.83E-01)	11.5	3.61E-01 (2.13E-01)	6.9	0.0	465.0	0.649855
GA	5.55E-01 (5.50E-01)	13.1	7.55E-05 (1.69E-04)	10.1	0.0	465.0	0.649855
ACO	1.72E+00 (7.28E-02)	34.1	8.80E-01 (2.01E-01)	10.6	0.0	465.0	0.649855
ES	1.39E+00 (2.83E-01)	8.5	6.53E-01 (2.28E-01)	7.8	0.0	465.0	0.649855
PBIL	1.04E+00 (3.04E-01)	6.6	5.19E-02 (8.06E-02)	4.9	0.0	465.0	0.649855
BP	9.00E-01 (6.14E-01)	12.7	3.02E+00 (6.11E-01)	0.7	465.0	0.0	0.000002

TABLE VII

EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM BALLOON

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	3.87E-19 (9.89E-21)	25.1	2.56E-34 (0.00E+00)	11.4	0.0	465.0	0.649855
PSO	2.45E-07 (9.31E-07)	24.4	2.98E-04 (7.82E-04)	12.2	460.0	5.0	0.000003
GA	3.66E-19 (2.13E-20)	26.0	1.11E-25 (3.56E-25)	13.1	0.0	465.0	0.649855
ACO	2.45E-07 (9.31E-07)	22.9	2.86E-01 (2.71E-01)	18.6	465.0	0.0	0.000002
ES	1.20E-04 (5.55E-04)	19.9	5.39E-02 (9.22E-02)	14.7	428.0	37.0	0.000033
PBIL	2.59E-10 (7.05E-11)	16.7	2.91E-05 (9.17E-05)	9.6	462.0	3.0	0.000002
BP	3.27E-02 (3.98E-03)	36.8	6.81E+00 (3.16E+00)	1.0	465.0	0.0	0.000002

equation

$$\frac{dx}{dt} = \beta \frac{x_\tau}{1 + x_\tau^n} - \gamma x \tag{15}$$

where β, γ , τ, and n are the real numbers, and x_τ denotes the value of variable x at time $t - \tau$. Box–Jenkins times series data are taken from [78]. EEG data are taken from http://www.cs.colostate.edu/eeg/main/data/1989_Keirn_and_Aunon. Note that only the mean-squared errors (MSE) of all tested problems are presented in this paper, while other evaluation metrics (e.g., classification accuracy [30]) can further be used for the specific problems.

As the primary purpose of this paper is to find out whether the proposed DNM performs better than the traditional MLP [54] and DNM with BP [29], we fix the settings of the parameters used in the learning algorithms. Alternatively, the sensitivities of user-defined parameters associated with DNM are analyzed. However, it should be noted that the

TABLE VIII
EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM IRIS

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	1.72E-18 (7.84E-34)	486.3	5.65E-39 (0.00E+00)	246.7	0.0	465.0	0.649855
PSO	2.87E-17 (7.79E-17)	495.0	1.78E-08 (3.29E-08)	273.0	465.0	0.0	0.000002
GA	1.65E-18 (2.31E-19)	500.3	6.59E-29 (3.39E-28)	256.4	0.0	465.0	0.649855
ACO	2.08E-09 (1.26E-24)	322.9	2.54E-03 (1.17E-02)	301.0	465.0	0.0	0.000002
ES	2.71E-18 (1.29E-19)	553.5	1.10E-05 (1.63E-05)	259.4	465.0	0.0	0.000002
PBIL	3.57E-18 (4.67E-18)	509.5	7.22E-12 (1.77E-11)	238.6	464.0	1.0	0.000002
BP	7.92E-02 (7.61E-03)	131.9	7.55E+01 (2.62E+01)	1.0	465.0	0.0	0.000002

TABLE IX
EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM BREAST CANCER

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	8.64E-02 (1.41E-17)	1626.7	1.67E-01 (5.65E-17)	1534.0	465.0	0.0	0.000002
PSO	4.49E-01 (2.21E-01)	1385.1	2.26E+00 (3.56E-01)	1567.6	465.0	0.0	0.000002
GA	1.36E-01 (5.87E-02)	1430.2	4.59E-01 (4.97E-01)	1538.4	368.0	97.0	0.005153
ACO	6.72E-01 (9.02E-02)	1357.0	1.08E+00 (2.24E-01)	1710.2	465.0	0.0	0.000002
ES	6.84E-01 (1.00E-01)	1270.4	2.71E+00 (4.96E-01)	1594.9	465.0	0.0	0.000002
PBIL	3.73E-01 (2.30E-01)	2031.2	2.12E+00 (4.02E-01)	1492.1	465.0	0.0	0.000002
BP	1.30E-01 (2.71E-02)	2323.9	7.29E+00 (1.76E+01)	3.4	465.0	0.0	0.000002

TABLE X
EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM HEART

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	3.50E+01 (1.45E-14)	305.9	4.67E+01 (1.45E-14)	321.8	465.0	0.0	0.000002
PSO	4.40E+01 (1.05E+01)	216.0	4.64E+01 (8.76E+00)	345.4	291.0	174.0	0.224926
GA	3.92E+01 (2.17E+00)	248.7	4.36E+01 (5.70E+00)	306.4	391.0	74.0	0.001074
ACO	4.24E+01 (3.68E+00)	415.7	4.88E+01 (8.56E+00)	443.6	409.0	56.0	0.000272
ES	4.66E+01 (4.38E+00)	222.5	5.10E+01 (7.48E+00)	329.7	334.0	131.0	0.035908
PBIL	4.02E+01 (3.73E+00)	203.8	4.66E+01 (6.21E+00)	275.6	420.0	45.0	0.00011
BP	3.57E+01 (7.13E+00)	176.6	9.05E+01 (2.37E+01)	2.4	465.0	0.0	0.000002

performance of learning algorithms can be significantly improved if we carefully select the values of their parameters through some preliminary experiments. To make a fair comparison with the reported data in the related research [54], the values listed in Table IV are used in our experiments, which are just the same as those in [54]. The population sizes of BBO, PSO, GA, ACO, ES, and PBIL are set to be 50 for XOR and balloon, and 200 for the rest problems. Their maximum number of generations is 250 whereas it in BP is set to be 3000 due to slow convergence speed. Other parameters are empirically set according to characteristics of respective algorithms.

TABLE XI
EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM SIGMOID

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	1.53E-02 (7.06E-18)	136.9	3.02E-02 (1.76E-17)	144.2	465.0	0.0	0.000002
PSO	5.23E-01 (5.81E-02)	157.2	1.52E+00 (3.16E-01)	962.1	465.0	0.0	0.000002
GA	3.55E-01 (3.06E-01)	163.2	1.53E-01 (2.24E-01)	912.5	60.0	405.0	0.427764
ACO	7.31E-01 (2.60E-01)	144.5	2.19E+00 (5.18E-01)	965.0	465.0	0.0	0.000002
ES	1.54E+00 (4.04E-01)	140.9	1.83E+00 (2.12E-01)	977.6	379.0	86.0	0.002498
PBIL	4.68E-01 (8.38E-03)	121.3	1.73E+00 (3.96E-01)	909.4	465.0	0.0	0.000002
BP	6.87E-01 (1.20E-02)	11.0	7.10E+00 (1.59E+01)	3.9	300.0	165.0	0.161919

TABLE XII
EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM COSINE

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	3.45E+01 (0.00E+00)	76.4	3.87E+01 (2.17E-14)	87.8	465.0	0.0	0.000002
PSO	3.30E+01 (3.44E+00)	102.0	3.87E+01 (3.00E+00)	109.3	441.0	24.0	0.000017
GA	3.46E+01 (4.77E-01)	96.5	3.90E+01 (8.84E-01)	94.3	465.0	0.0	0.000002
ACO	3.43E+01 (2.69E+00)	88.1	3.97E+01 (3.40E+00)	114.7	432.0	33.0	0.000037
ES	3.45E+01 (8.59E+00)	82.8	3.88E+01 (5.63E+00)	92.6	368.0	97.0	0.005153
PBIL	3.40E+01 (4.62E+00)	74.2	3.90E+01 (3.17E+00)	82.6	401.0	64.0	0.000509
BP	4.07E+01 (1.42E+01)	3.4	4.23E+01 (1.68E+01)	0.7	286.0	179.0	0.266702

TABLE XIII
EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM SINE

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	1.21E+02 (0.00E+00)	251.0	1.39E+02 (1.16E-13)	268.9	465.0	0.0	0.000002
PSO	1.42E+02 (6.20E-01)	269.5	1.61E+02 (2.17E+01)	294.5	414.0	51.0	0.000182
GA	1.32E+02 (3.61E+01)	281.4	1.53E+02 (7.76E+00)	272.3	409.0	56.0	0.000272
ACO	1.29E+02 (2.75E+01)	263.8	1.69E+02 (3.03E+01)	304.2	410.0	55.0	0.000251
ES	1.50E+02 (4.69E+01)	234.8	1.75E+02 (2.23E+01)	279.7	347.0	118.0	0.018013
PBIL	1.24E+02 (2.38E+01)	239.3	1.60E+02 (2.31E+01)	263.9	436.0	29.0	0.000027
BP	1.82E+02 (2.19E+01)	9.3	1.58E+02 (6.55E+01)	1.0	94.0	371.0	0.21863

B. Sensitivity Analysis of User-Defined Parameters

In order to obtain the best performance for DNM, its user-defined parameters are worth investigating. There are four key parameters in DNM, i.e., the number (M) of dendrites in a model, the synaptic parameter (k) in the connection sigmoid function, and two soma parameters (k_s and θ_s) in the output sigmoid function. It is obvious that M remarkably influences the computational time of DNM.

In this experiment, the Taguchi's method [57] is used to acquire a reasonable combination of four DNM parameters. It detects a part of the possible combinations among factors rather than the whole combinations, thus resulting

TABLE XIV

EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM SPHERE

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	1.38E+04 (8.23E+01)	1074.6	1.44E+04 (0.00E+00)	897.5	465.0	0.0	0.000002
PSO	1.21E+04 (5.55E+02)	1037.4	1.47E+04 (6.99E+02)	923.8	465.0	0.0	0.000002
GA	1.39E+04 (1.28E+03)	1132.5	1.44E+04 (3.02E+02)	881.3	301.0	164.0	0.154773
ACO	1.14E+04 (6.59E+02)	1165.1	1.53E+04 (7.77E+02)	907.2	465.0	0.0	0.000002
ES	1.14E+04 (2.69E+02)	1087.6	1.52E+04 (8.30E+02)	933.3	465.0	0.0	0.000002
PBIL	1.33E+04 (5.98E+02)	1103.3	1.47E+04 (7.77E+02)	875.1	457.0	8.0	0.000004
BP	1.07E+04 (0.00E+00)	380.4	1.46E+04 (1.19E+03)	2.3	465.0	0.0	0.000002

TABLE XV

EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM GRIEWANK

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	4.78E+02 (2.31E-13)	960.5	6.96E+05 (0.00E+00)	899.8	465.0	0.0	0.000002
PSO	5.08E+02 (6.15E+01)	1047.9	6.93E+05 (5.07E+03)	923.3	465.0	0.0	0.000002
GA	5.38E+02 (9.59E+01)	961.8	6.92E+05 (1.90E+03)	896.8	465.0	0.0	0.000002
ACO	5.58E+02 (4.23E+01)	928.0	6.92E+05 (6.74E+03)	917.0	465.0	0.0	0.000002
ES	5.30E+02 (8.37E+01)	1117.6	6.89E+05 (4.93E+03)	934.8	465.0	0.0	0.000002
PBIL	5.27E+02 (2.76E+01)	842.3	6.94E+05 (3.89E+03)	876.0	465.0	0.0	0.000002
BP	4.29E+02 (3.50E+01)	380.6	6.95E+05 (7.00E+03)	2.3	465.0	0.0	0.000002

TABLE XVI

EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM ROSENBROCK

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	2.98E+07 (2.01E+04)	3190.2	3.00E+07 (1.26E+05)	2439.1	465.0	0.0	0.000002
PSO	2.99E+07 (1.24E+05)	2414.9	2.99E+07 (1.33E+05)	2466.4	308.0	157.0	0.116119
GA	2.99E+07 (1.35E+05)	2533.9	2.99E+07 (1.08E+05)	2406.8	358.0	107.0	0.009008
ACO	2.99E+07 (8.26E+04)	3381.1	3.00E+07 (1.80E+05)	2405.1	390.0	75.0	0.001155
ES	2.99E+07 (8.63E+04)	3176.9	3.00E+07 (1.14E+05)	2495.8	397.0	68.0	0.000689
PBIL	2.99E+07 (1.27E+05)	2389.0	3.00E+07 (1.15E+05)	2353.1	356.0	109.0	0.010556
BP	2.95E+07 (1.22E+05)	2181.8	2.98E+07 (9.39E+04)	3.9	465.0	0.0	0.000002

in a minimum experimental run and the best estimation of factors during the execution [79]. The number of levels for four factors are set as follows: five levels for $M \in \{1, 3, 5, 10, 20\}$; five levels for $k \in \{1, 5, 10, 15, 25\}$; five levels for $k_s \in \{1, 5, 10, 15, 25\}$; and five levels for $\theta_s \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. A full-factorial analysis needs $5^4 = 625$ experiments. Compared with the full-factorial analysis, Taguchi's method adopts the orthogonal arrays so as to decrease greatly the number of experimental runs, the cost of time, manpower, and materials. Therefore, an orthogonal array $L_{25}(5^4)$ that contains only 25 experiments is adopted in our experiment.

TABLE XVII

EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM MACKEY GLASS

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	4.65E-01 (2.54E-06)	885.6	1.77E+00 (6.78E-16)	928.9	465.0	0.0	0.000002
PSO	1.14E+00 (2.30E-01)	914.3	1.09E+01 (2.47E+00)	962.1	465.0	0.0	0.000002
GA	1.21E+00 (6.14E-01)	948.4	1.83E+00 (1.07E+00)	912.5	343.0	122.0	0.022425
ACO	2.65E+00 (7.23E-01)	976.3	2.14E+01 (6.92E+00)	965.0	465.0	0.0	0.000002
ES	3.04E+00 (1.17E+00)	920.0	1.97E+01 (5.49E+00)	977.6	465.0	0.0	0.000002
PBIL	1.34E+00 (2.61E-01)	870.3	1.08E+01 (2.68E+00)	909.4	465.0	0.0	0.000002
BP	8.84E-01 (7.80E-02)	512.5	2.77E+02 (1.71E+02)	3.9	465.0	0.0	0.000002

TABLE XVIII

EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM BOX JENKINS

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	1.40E+00 (2.26E-16)	307.8	1.24E+00 (6.78E-16)	304.9	0.0	465.0	0.649855
PSO	1.28E+00 (1.60E-01)	321.8	3.16E+00 (7.18E-01)	334.6	465.0	0.0	0.000002
GA	1.60E+00 (2.75E-01)	321.3	1.30E+00 (2.61E-01)	305.6	45.0	420.0	0.51650
ACO	1.50E+00 (3.50E-02)	313.7	4.79E+00 (1.49E+00)	338.5	465.0	0.0	0.000002
ES	1.66E+00 (2.29E-01)	284.4	5.90E+00 (1.50E+00)	322.2	465.0	0.0	0.000002
PBIL	1.20E+00 (2.46E-02)	290.7	2.96E+00 (7.24E-01)	296.5	465.0	0.0	0.000002
BP	1.39E+00 (1.42E-02)	81.3	5.66E+01 (2.36E+01)	2.3	465.0	0.0	0.000002

TABLE XIX

EXPERIMENTAL RESULTS OF MLP AND DNM FOR THE PROBLEM EEG

Algorithm	Results of DNM		Results of MLP		R^+	R^-	p-value
	AVE (STD)	Time	AVE (STD)	Time			
BBO	1.39E+01 (5.42E-15)	1918.5	1.46E+01 (5.42E-15)	2019.1	465.0	0.0	0.000002
PSO	1.57E+01 (6.67E-01)	1949.1	3.50E+01 (6.46E+00)	2059.5	465.0	0.0	0.000002
GA	1.53E+01 (7.58E-01)	2118.8	1.63E+01 (1.51E+00)	1983.1	372.5	92.5	0.003764
ACO	1.79E+01 (1.97E+00)	1804.7	4.48E+01 (4.97E+00)	2213.1	465.0	0.0	0.000002
ES	2.04E+01 (3.15E+00)	1912.3	4.61E+01 (4.22E+00)	2120.0	465.0	0.0	0.000002
PBIL	1.59E+01 (4.75E-01)	1922.2	3.35E+01 (6.06E+00)	1982.3	465.0	0.0	0.000002
BP	1.42E+01 (1.31E-01)	1299.7	3.08E+02 (1.81E+02)	7.9	465.0	0.0	0.000002

Tables I–XIV in Supplementary Material summarize the experimental results where MSE indicates mean square error values with 30 runs of the seven learning algorithms (i.e., BBO, PSO, GA, ACO, ES, PBIL, and BP) for 14 tested problems. As a result, we obtain acceptable user-defined parameter settings according to our experimental results for each problem independently, summarized in Table V.

C. Performance Comparison

Tables VI–XIX summarize the results of MLP and DNM for the 14 problems under the seven learning algorithms. Wilcoxon signed-ranks test [80] is used to detect significant

TABLE XX

OVERALL STATISTICAL COMPARISON OBTAINED BY FRIEDMAN TEST OF MLP AND DNM WITH SEVEN LEARNING ALGORITHMS FOR 14 DIFFERENT PROBLEMS

Algorithm	Average ranking	Unadjusted p	p_{Bonf}	p_{Holm}	$p_{Hochberg}$	$\alpha = 0.05$
BBO+DNM	2.6071	-	-	-	-	-
PBIL+DNM	4.5357	0.222565	1	0.297503	0.222565	No
GA+DNM	4.9643	0.136017	1	0.297503	0.222565	No
PSO+DNM	5.2143	0.099168	1	0.297503	0.222565	No
BP+DNM	5.9286	0.035671	0.463725	0.159432	0.142685	Yes
BBO+MLP	6	0.031886	0.414524	0.159432	0.142685	Yes
GA+MLP	6.2857	0.01999	0.259872	0.119941	0.119941	Yes
ACO+DNM	6.7857	0.008223	0.106903	0.057563	0.057563	Yes
ES+DNM	7.6429	0.001448	0.018827	0.011586	0.011586	Yes
PBIL+MLP	9.6071	0.00001	0.000124	0.000086	0.000086	Yes
PSO+MLP	9.6429	0.000009	0.000112	0.000086	0.000086	Yes
ES+MLP	11.5357	0	0	0	0	Yes
ACO+MLP	11.7857	0	0	0	0	Yes
BP+MLP	12.4643	0	0	0	0	Yes

differences between the behaviors of two algorithms. In Tables VI–XIX, R^+ denotes the sum of ranks for the problems in which the control algorithm outperforms the competitive one, and R^- indicates the sum of ranks for the opposite. A p-value smaller than 0.05 signifies that DNM performs significantly better than MLP under the same learning algorithm. In these tables, DNM is the control algorithm. From these tables, it can be found that in 79 out of the whole 98 cases, DNM performs better than MLP, suggesting that the proposed DNM is a promising neuron computation tool for classification, function approximation, and prediction tasks.

Table XX shows the overall statistical results of MLP and DNM with seven learning algorithms for 14 problems via the Friedman test at the level of $\alpha = 0.05$ with three *post-hoc* procedures. *Post-hoc* procedures are able to properly compare several algorithms and enhance the accuracy of statistical results, which can calculate an adjusted p-value to judge the degree of rejection of each hypothesis. In our Friedman test, three common *post-hoc* procedures, i.e., the Bonferroni–Dunn procedure, Holm procedure, and Hochberg procedure, are used to obtain different adjusted p-values shown as p_{Bonf}, p_{Holm}, and p_{Hochberg} in Table XX according to their distinctive calculation mechanisms that change the value of significance α in different ways. Compared with an unadjusted p-value, an adjusted p-value considers the family error accumulated and adjusts the value of α to give more accurate information for multiple comparisons. From this table, it is observed that DNM with BBO performs the best among all the algorithms, and majority of DNMs with learning algorithms outperform MLP with those according to the average ranking, indicating DNMs with learning algorithms are more effective. Moreover, DNM with BBO is significantly superior to MLP with seven learning algorithms on the basis of the unadjusted p, suggesting that BBO is the most effective method for training DNM to obtain desired solutions. It is interesting that the MLP with BBO also performs remarkably better than the six other learning algorithms, thereby suggesting the superiority of BBO for training both DNM and MLP. Finally, the DNM with BBO generally exhibits the best performance for classification, approximation, and prediction problems.

In addition, Figs. 1–14 in Supplementary Material depict the solution distribution of 30 independent runs of different learning algorithms in MLP and DNM for all tested problems. From these figures, a clear observation regarding the average MSE and the robustness of learning algorithms can be made. BBO usually finds stable solutions over different runs, and on the contrary, BP often varies much among different runs. Furthermore, Figs. 15–28 in Supplementary File illustrate the convergence graphs of BBO, PSO, GA, ACO, ES, PBIL, and BP in MLP and DNM for all tested problems. It can be easily found that the learning algorithms generally converge faster than BP. In most cases, BBO converges the fastest and can find the minimal MSE for the tested problems.

V. CONCLUSION

In this paper, a novel DNM by considering the nonlinearity of synapses is used for resolving various optimization problems. The conventional DNM uses a BP learning method to train it. However, BP is a limited learning algorithm because of its inherent local-minimum trapping problem and usually fails to find the global best values for DNM weights and thresholds. Thus, DNM performance is limited.

To achieve its better performance, six learning algorithms including BBO, PSO, GA, ACO, ES, and PBIL are for the first time adopted to train DNM in this paper. A set of 14 problems, i.e., N-bit XOR classification, balloon classification, iris classification, breast cancer classification, heart classification, sigmoid function approximation, cosine function approximation, sine function approximation, sphere function approximation, Griewank function approximation, Rosenbrock function approximation, Mackey–Glass prediction, Box–Jenkins chaotic data prediction, and EEG data prediction, are adopted to test the DNM performance.

The best combination of user-defined DNM parameters (i.e., the number of dendrites, the synapse connection parameter, and two soma parameters) has been systemically investigated and determined by using the Taguchi's experimental design method. Statistical analysis in terms of average values of the sum of mean square errors, standard deviation, and nonparametric test results are thoroughly performed. The results suggest that the tested learning algorithms are effective and promising for training both DNM and MLP. We conclude that BBO is the most competitive for enhancing the performances of DNM and MLP among the seven learning algorithms, and

DNM with these learning algorithms mostly outperforms MLP with the same algorithms. Hence, this work has established DNM with six learning algorithms, especially as a powerful tool in solving classification, approximation, and prediction problems. In the future work, the DNM with BBO should be further explored and developed to solve other different problems [81]–[84]. More advanced algorithms [73], [83]–[85] should be combined with it to improve its property.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[3] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.

[4] S. Haykin, *Neural Networks and Learning Machines*, vol. 3. Upper Saddle River, NJ, USA: Pearson, 2009.

[5] Z. Zhang, Z. Li, Y. Zhang, Y. Luo, and Y. Li, "Neural-dynamic-method-based dual-arm CMG scheme with time-varying constraints applied to humanoid robots," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3251–3262, Dec. 2015.

[6] Z. Zhang, L. Zheng, J. Yu, Y. Li, and Z. Yu, "Three recurrent neural networks and three numerical methods for solving a repetitive motion planning scheme of redundant robot manipulators," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 3, pp. 1423–1434, Jun. 2017.

[7] R. N. Yadav, P. K. Kalra, and J. John, "Time series prediction with single multiplicative neuron model," *Appl. Soft Comput.*, vol. 7, no. 4, pp. 1157–1163, Aug. 2007.

[8] L. Zhao and Y. Yang, "PSO-based single multiplicative neuron model for time series prediction," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 2805–2812, 2009.

[9] Q. Kang, B. Huang, and M. Zhou, "Dynamic behavior of artificial Hodgkin–Huxley neuron model subject to additive noise," *IEEE Trans. Cybern.*, vol. 46, no. 9, pp. 2083–2093, Sep. 2016.

[10] R. Legenstein and W. Maass, "Branch-specific plasticity enables self-organization of nonlinear computation in single neurons," *J. Neurosci.*, vol. 31, no. 30, pp. 10787–10802, 2011.

[11] J. Bono, K. A. Wilmes, and C. Clopath, "Modelling plasticity in dendrites: From single cells to networks," *Current Opinion Neurobiol.*, vol. 46, pp. 136–141, Oct. 2017.

[12] M. Schiess, R. Urbanczik, and W. Senn, "Somato-dendritic synaptic plasticity and error-backpropagation in active dendrites," *PLoS Comput. Biol.*, vol. 12, no. 2, p. e1004638, 2016.

[13] K. N. Gurney, "Training nets of hardware realizable sigma-pi units," *Neural Netw.*, vol. 5, no. 2, pp. 289–303, 1992.

[14] B. Lenze, "How to make sigma-pi neural networks perform perfectly on regular training sets," *Neural Netw.*, vol. 7, no. 8, pp. 1285–1293, 1994.

[15] M. Heywood and P. Noakes, "A framework for improved training of sigma-pi networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 4, pp. 893–903, Jul. 1995.

[16] R. S. Neville and S. Eldridge, "Transformations of sigma-pi nets: Obtaining reflected functions by reflecting weight matrices," *Neural Netw.*, vol. 15, no. 3, pp. 375–393, 2002.

[17] C. Zhang, W. Wu, and Y. Xiong, "Convergence analysis of batch gradient algorithm for three classes of sigma-pi neural networks," *Neural Process. Lett.*, vol. 26, no. 3, pp. 177–189, 2007.

[18] J. Long, W. Wu, and D. Nan, "L^p approximation capabilities of sum-of-product and sigma-pi–Sigma neural networks," *Int. J. Neural Syst.*, vol. 17, no. 5, pp. 419–424, 2007.

[19] C. Weber and S. Wermter, "A self-organizing map of sigma-pi units," *Neurocomputing*, vol. 70, nos. 13–15, pp. 2552–2560, 2007.

[20] C. Koch and I. Segev, "The role of single neurons in information processing," *Nature Neurosci.*, vol. 3, pp. 1171–1177, Nov. 2000.

[21] R. P. Costa and P. J. Sjöström, "One cell to rule them all, and in the dendrites bind them," *Frontiers Synaptic Neurosci.*, vol. 3, Sep. 2011, Art. no. 5.

[22] C. Koch, T. Poggio, and V. Torre, "Nonlinear interactions in a dendritic tree: Localization, timing, and role in information processing," *Proc. Nat. Acad. Sci. USA*, vol. 80, no. 9, pp. 2799–2802, 1983.

[23] N. Brunel, V. Hakim, and M. J. Richardson, "Single neuron dynamics and computation," *Current Opinion Neurobiol.*, vol. 25, pp. 149–155, Apr. 2014.

[24] R. D. Cazé, S. Jarvis, A. J. Foust, and S. R. Schultz, "Dendrites enable a robust mechanism for neuronal stimulus selectivity," *Neural Comput.*, vol. 29, no. 9, pp. 2511–2527, 2017.

[25] L. F. Abbott and W. G. Regehr, "Synaptic computation," *Nature*, vol. 431, pp. 796–803, Oct. 2004.

[26] A. Destexhe and E. Marder, "Plasticity in single neuron and circuit computations," *Nature*, vol. 431, no. 7010, pp. 789–795, 2004.

[27] Y.-N. Jan and L. Y. Jan, "Branching out: Mechanisms of dendritic arborization," *Nature Rev. Neurosci.*, vol. 11, no. 5, pp. 316–328, 2010.

[28] Y. Todo, H. Tamura, K. Yamashita, and Z. Tang, "Unsupervised learnable neuron model with nonlinear interaction on dendrites," *Neural Netw.*, vol. 60, pp. 96–103, Dec. 2014.

[29] Z. Sha, L. Hu, Y. Todo, J. Ji, S. Gao, and Z. Tang, "A breast cancer classifier using a neuron model with dendritic nonlinearity," *IEICE Trans. Inf. Syst.*, vol. E98-D, no. 7, pp. 1365–1376, 2015.

[30] J. Ji, S. Gao, J. Cheng, Z. Tang, and Y. Todo, "An approximate logic neuron model with a dendritic structure," *Neurocomputing*, vol. 173, pp. 1775–1783, Jan. 2016.

[31] T. Jiang, S. Gao, D. Wang, J. Ji, Y. Todo, and Z. Tang, "A neuron model with synaptic nonlinearities in a dendritic tree for liver disorders," *IEEJ Trans. Elect. Electron. Eng.*, vol. 12, no. 1, pp. 105–115, 2017.

[32] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, and Z. Tang, "Financial time series prediction using a dendritic neuron model," *Knowl.-Based Syst.*, vol. 105, pp. 214–224, Aug. 2016.

[33] W. Chen, J. Sun, S. Gao, J.-J. Cheng, J. Wang, and Y. Todo, "Using a single dendritic neuron to forecast tourist arrivals to Japan," *IEICE Trans. Inf. Syst.*, vol. E100-D, no. 1, pp. 190–202, 2017.

[34] B. W. Mel and C. Koch, "Sigma-pi learning: On radial basis functions and cortical associative learning," in *Proc. NIPS*, 1989, pp. 474–481.

[35] M. P. Jadi, B. F. Behabadi, A. Poleg-Polsky, J. Schiller, and B. W. Mel, "An augmented two-layer model captures nonlinear analog spatial integration effects in pyramidal neuron dendrites," *Proc. IEEE*, vol. 102, no. 5, pp. 782–798, May 2014.

[36] F. Gabbiani, H. G. Krapp, C. Koch, and G. Laurent, "Multiplicative computation in a visual neuron sensitive to looming," *Nature*, vol. 420, no. 6913, pp. 320–324, 2002.

[37] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, Sep. 1990.

[38] S. Ruder. (2016). "An overview of gradient descent optimization algorithms." [Online]. Available: https://arxiv.org/abs/1609.04747

[39] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, Mar./Apr. 1993, pp. 586–591.

[40] C. Igel and M. Hüsken, "Empirical evaluation of the improved RPROP learning algorithms," *Neurocomputing*, vol. 50, pp. 105–123, Jan. 2003.

[41] A. C. Veitch and G. Holmes, "A modified quickprop algorithm," *Neural Comput.*, vol. 3, no. 3, pp. 310–311, 1991.

[42] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2933–2941.

[43] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 54–65, Jan. 1994.

[44] R. S. Sexton, R. E. Dorsey, and J. D. Johnson, "Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation," *Decis. Support Syst.*, vol. 22, no. 2, pp. 171–185, 1998.

[45] J.-R. Zhang, J. Zhang, T.-M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training," *Appl. Math. Comput.*, vol. 185, no. 2, pp. 1026–1037, 2007.

[46] H. A. Abbass, "Speeding up backpropagation using multiobjective evolutionary algorithms," *Neural Comput.*, vol. 15, no. 11, pp. 2705–2726, 2003.

[47] R. S. Sexton and J. N. D. Gupta, "Comparative evaluation of genetic algorithm and backpropagation for training neural networks," *Inf. Sci.*, vol. 129, nos. 1–4, pp. 45–59, 2000.

[48] Y. Q. Zhang, B. Jin, and Y. Tang, "Granular neural networks with evolutionary interval learning," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 2, pp. 309–319, Apr. 2008.

[49] D. Golmohammadi, R. C. Creese, H. Valian, and J. Kolassa, "Supplier selection based on a neural network model using genetic algorithm," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1504–1519, Sep. 2009.

[50] S. C. Tan, J. Watada, Z. Ibrahim, and M. Khalid, "Evolutionary fuzzy ARTMAP neural networks for classification of semiconductor defects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 933–950, May 2015.

[51] L. Zhang and P. N. Suganthan, "A survey of randomized algorithms for training neural networks," *Inf. Sci.*, vols. 364–365, pp. 146–155, Oct. 2016.

[52] F. Gaxiola, P. Melin, F. Valdez, J. R. Castro, and O. Castillo, "Optimization of type-2 fuzzy weights in backpropagation learning for neural networks using GAs and PSO," *Appl. Soft Comput.*, vol. 38, pp. 860–871, Jan. 2016.

[53] B. Jafrasteh and N. Fathianpour, "A hybrid simultaneous perturbation artificial bee colony and back-propagation algorithm for training a local linear radial basis neural network on ore grade estimation," *Neurocomputing*, vol. 235, pp. 217–227, Apr. 2017.

[54] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Let a biogeography-based optimizer train your multi-layer perceptron," *Inf. Sci.*, vol. 269, pp. 188–209, Jun. 2014.

[55] A. P. Piotrowski, "Differential evolution algorithms applied to neural network training suffer from stagnation," *Appl. Soft Comput.*, vol. 21, pp. 382–406, Aug. 2014.

[56] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[57] G. Taguchi, R. Jugulum, and S. Taguchi, *Computer-Based Robust Engineering: Essentials for DFSS*. Mexico, North America: ASQ Quality Press, 2004.

[58] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.

[59] L. Grippo, A. Manno, and M. Sciandrone, "Decomposition techniques for multilayer perceptron training," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 11, pp. 2146–2159, Nov. 2016.

[60] K. Sun, S.-H. Huang, D. S.-H. Wong, and S.-S. Jang, "Design and application of a variable selection method for multilayer perceptron neural network with LASSO," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1386–1396, Jun. 2017.

[61] R. Chakraborty and N. R. Pal, "Feature selection using a neural framework with controlled redundancy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 35–50, Jan. 2015.

[62] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.

[63] D. Simon, *Evolutionary Optimization Algorithms: Biologically-Inspired and Population-Based Approaches to Computer Intelligence*. Hoboken, NJ, USA: Wiley, 2014.

[64] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci.*, vol. 1, New York, NY, USA, Oct. 1995, pp. 39–43.

[65] V. G. Gudise and G. K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks," in *Proc. IEEE Swarm Intell. Symp. (SIS)*, Apr. 2003, pp. 110–117.

[66] M. Meissner, M. Schmuker, and G. Schneider, "Optimized particle swarm optimization (OPSO) and its application to artificial neural network training," *BMC Bioinf.*, vol. 7, no. 1, p. 125, 2006.

[67] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[68] S. Kiranyaz, T. Ince, A. Yildirim, and M. Gabbouj, "Evolutionary artificial neural networks by multi-dimensional particle swarm optimization," *Neural Netw.*, vol. 22, no. 10, pp. 1448–1462, 2009.

[69] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[70] J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2390–2401, Nov. 2015.

[71] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1155–1173, 2008.

[72] A. P. Engelbrecht, *Computational Intelligence: An Introduction*. Hoboken, NJ, USA: Wiley, 2007.

[73] 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.

[74] P. Larrañaga, "A review on estimation of distribution algorithms," in *Estimation of Distribution Algorithms*. Boston, MA, USA: Springer, 2002, pp. 57–100.

[75] C. González, J. A. Lozano, and P. Larrañaga, "Analyzing the population based incremental learning algorithm by means of discrete dynamical systems," *Complex Syst.*, vol. 12, no. 4, pp. 465–479, 2000.

[76] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana, "A review on probabilistic graphical models in evolutionary computation," *J. Heuristics*, vol. 18, no. 5, pp. 795–819, 2012.

[77] C. Blake. (1998). *UCI Repository of Machine Learning Databases*. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[78] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[79] J. F. C. Khaw, B. S. Lim, and L. E. N. Lim, "Optimal design of neural networks using the Taguchi method," *Neurocomputing*, vol. 7, no. 3, pp. 225–245, 1995.

[80] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization," *J. Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

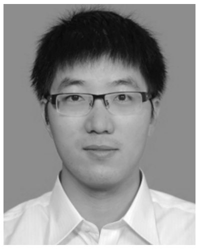
[81] C. Li, L. Wang, G. Zhang, H. Wang, and F. Shang, "Functional-type single-input-rule-modules connected neural fuzzy system for wind speed prediction," *IEEE/CAA J. Autom. Sin.*, vol. 4, no. 4, pp. 751–762, Apr. 2017.

[82] S. Teng, N. Wu, H. Zhu, L. Teng, and W. Zhang, "SVM-DT-based adaptive and collaborative intrusion detection," *IEEE/CAA J. Autom. Sin.*, vol. 5, no. 1, pp. 108–118, Jan. 2018.

[83] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 524–537, Mar. 2016.

[84] W. Han *et al.*, "Cuckoo search and particle filter-based inversing approach to estimating defects via magnetic flux leakage signals," *IEEE Trans. Magn.*, vol. 52, no. 4, Apr. 2016, Art. no. 6200511.

[85] H. Yuan, J. Bi, W. Tan, M. C. Zhou, B. H. Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.



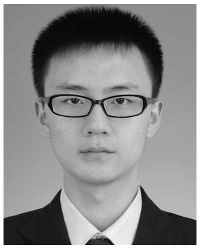
Shangce Gao (M'11–SM'16) received the Ph.D. degree in innovative life science from the University of Toyama, Toyama, Japan, in 2011. He is currently an Associate Professor with the Faculty of Engineering, University of Toyama. He has authored over 90 publications in referred journals and conference proceedings. His current research interests include computational intelligence and its applications.

Dr. Gao was a recipient of the Best Paper Award at the IEEE International Conference on Progress in Informatics and Computing, the Shanghai Rising-Star Scientist Award, the Chen-Guang Scholar of Shanghai Award, the Outstanding Academic Performance Award of IEICE, and the Outstanding Academic Achievement Award of IPSJ.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

In 1990, he joined the New Jersey Institute of Technology, Newark, NJ, USA, where he is currently a Distinguished Professor of electrical and computer engineering. He has authored or co-authored over 700 publications including 12 books, more than 400 journal papers (over 300 in the IEEE Transactions), and 28 book chapters. His current research interests include Petri nets, intelligent automation, Internet of Things, big data, web services, and intelligent transportation.

Dr. Zhou is a Life Member and the President of the Chinese Association for Science and Technology, USA, in 1999. He is a Fellow of the International Federation of Automatic Control and the American Association for the Advancement of Science. He was invited to lecture in Australia, Canada, China, France, Germany, Hong Kong, Italy, Japan, South Korea, Mexico, Saudi Arabia, Singapore, Taiwan, and U.S. and served as a plenary/keynote speaker for many conferences. He was a recipient of the Humboldt Research Award for U.S. Senior Scientists from Alexander von Humboldt Foundation, the Franklin V. Taylor Memorial Award, and the Norbert Wiener Award from the IEEE Systems, Man, and Cybernetics Society. He is the Founding Editor of *IEEE Press Book Series on Systems Science and Engineering*, an Editor-in-Chief of the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the IEEE INTERNET OF THINGS JOURNAL, and the *Frontiers of Information Technology & Electronic Engineering*.

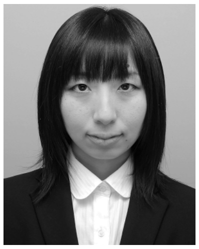


Yirui Wang received the B.S. and M.S. degrees from the College of Information Sciences and Technology, Donghua University, Shanghai, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the University of Toyama, Toyama, Japan.

His current research interests include computational intelligence, swarm intelligent algorithms, and combinatorial optimizations.

Jiujun Cheng received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2006.

In 2009, he was a Visiting Professor with Aalto University, Espoo, Finland. He is currently a Professor with Tongji University, Shanghai, China. He has authored or co-authored over 40 publications including conference and journal papers. His current research interests include mobile computing, social network with a focus on mobile/Internet interworking, service computing, and Internet of Vehicles.



Hanaki Yachi received the B.S. and M.S. degrees from the University of Toyama, Toyama, Japan, in 2014 and 2017, respectively, where she is currently pursuing the Ph.D. degree.

Her current research interests include neural networks and its applications.

Jiahai Wang (M'07) received the Ph.D. degree from the University of Toyama, Toyama, Japan, in 2005.

In 2005, he joined Sun Yat-sen University, Guangzhou, China, where he is currently a Professor. His current research interests include computational intelligence and its applications.