Received March 11, 2019, accepted March 30, 2019, date of publication April 16, 2019, date of current version April 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911530

Large-Scale Feedforward Neural Network Optimization by a Self-Adaptive Strategy and Parameter Based Particle Swarm Optimization

YU XUE[1,2], (Member, IEEE), TAO TANG[1], AND ALEX X. LIU[2], (Fellow, IEEE)
[1]School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China
[2]Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA

Corresponding author: Yu Xue (xueyu@nuist.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61403206, Grant 61876089, and Grant 61876185, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20141005, in part by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant 14KJB520025, in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions, in part by the Open Research Fund of Jiangsu Engineering Research Center of Communication and Network Technology, and in part by Nanjing University of Posts and Telecommunications (NJUPT) under Grant JSGCZX17001.

ABSTRACT Feedforward neural network (FNN) is one of the most widely used and fastest-developed artificial neural networks. Much evolutionary computation (EC) methods have been used to optimize the weights of FNN. However, as the dimension of datasets increases, the number of weights also increases dramatically. On high-dimensional datasets, if EC methods are used directly to optimize the weights of FNN, it is impossible to obtain the optimal weights of the FNN by EC methods in an acceptable time. Feature selection is a method that can effectively reduce the computational complexity of FNN by reducing irrelevant and redundant features. It may be practical to optimize the FNN by EC methods if we first employ the feature selection for the large-scale datasets. In this paper, we present a self-adaptive parameter and strategy-based particle swarm optimization (SPS-PSO) algorithm to optimize FNN with feature selection. First, we propose an optimization model for FNN by transforming the designing of FNN into a weights optimization problem. Simultaneously, we present a feature selection optimization model. Second, we present an SPS-PSO algorithm. In this algorithm, we use the parameter and strategy self-adaptive mechanism. In addition, five candidate solution generating strategies (CSGS) are used. The experiments are divided into two groups. In the first group, SPS-PSO and three other EC methods are used to directly optimize the weights of FNN on eight datasets without any modification. In the second group, we first employ SPS-PSO-based feature selection on the original datasets and obtain eight relatively smaller datasets with the *k*-nearest neighbor (KNN) which is used as the evaluation function for saving time. Then, we use the new datasets as the inputs for FNN. We optimize the weights of FNN again by SPS-PSO and three other EC methods to investigate whether we can get similar or even better classification accuracy by comparing the results with that of the first group. The experimental results show that SPS-PSO has the advantage in optimizing the weights of FNN compared with the other EC methods. Meanwhile, the SPS-PSO-based feature selection can reduce the solution size and computational complexity while ensuring the classification accuracy when it is used to preprocess the datasets for FNN. In this method, a solution with an originally higher than 700 000 dimensions can be even reduced to hundreds of dimensions.

INDEX TERMS Classification, evolutionary computation, feature selection, feedforward neural networks, self-adaptive, parameter adaptation, particle swarm optimization.

I. INTRODUCTION

Artificial neural network (ANN) [1], [2] is a hot research topic in the field of artificial intelligence [3] since the 1980s and it has been widely used as an effective classification method. ANN is a nonlinear and adaptive information processing system composed of a large number of interconnected processing units. It abstracts the human brain neuron network from the perspective of information processing, establishes

The associate editor coordinating the review of this manuscript and approving it for publication was Yongming Li.

a simple model, and forms different networks according to different connection methods. ANN mainly contains the feedforward neural network (FNN) [4] and the feedback neural network (FeedbackNN) [5]. FNN has the characteristics of simple network structure and easy implementation, hence it has been widely studied by researchers in the past few decades and is one of the most widely used and fastest-developing ANN. So far, FNN has been applied in spectrum auction [6], medicine [7], direction prediction [8], construction engineering [9], instruction detection [10], [11] and etc.

Recently, many EC methods [12] have been used to optimize the weights for FNN [13], [14]. Gudise and Venayagamoorthy [15] employed particle swarm optimization (PSO) to the FNN weights optimization and compared the results with back propagation (BP) technique. The results show that PSO has the advantage on searching the optimal weights for FNN compared with BP technique. Lam *et al.* [16] used an improved genetic algorithm (GA) to optimize the weights of FNN and achieved good results. Biogeography-based optimization [17] was introduced into the FNN weights optimization by Zhang *et al.*, and they use the optimized FNN to solve fruit classification problems. In [18], Bohat and Arya studied the existing gravitational search algorithm (GSA) and further proposed a novel algorithm named gbest-guided gravitational search algorithm (GG-GSA). This algorithm is used to optimize the weights of FNN and also achieves a good performance on classification accuracy compared with GSA and some other EC methods. However, which is the same as the study of stochastic optimization problems in most literatures, the datasets chosen in [18] are all small-scale datasets, the biggest one of them only has 41 features while the corresponding number of weights (solution size of EC methods) is 3570. In other words, the researcher did not use GG-GSA to solve large-scale FNN weights optimization problems. In high-dimensional datasets, oversized solutions produce extremely huge solution space, it takes more time to search for the optimal solution, and the sharply increasing irrelevant and redundant features can bring more local optima to the FNN optimization problems. For a dataset containing n features, the solution size of the FNN optimization problem can be greater than $2n^2$. For example, if a dataset originally contains 5000 features, the solution size of the FNN optimization problem is more than 50,000,000. If the dimension of the dataset is extremely high, due to hardware limitations, ordinary personal computers even can not run the EC methods with a extremely large solution size or obtain results in an acceptable time. Therefore, it becomes impossible to optimize the FNN optimization problems by the EC methods if the datasets are extremely large, so it is urgent to find a new technique to solve this problem.

As an effective data preprocessing technique, feature selection [19], [20] can effectively reduce the irrelevant or redundant features in datasets [21]. Thus, by introducing the feature selection method, it may be possible to optimize the weights of the FNN by EC methods on extremely large-scale datasets. When feature selection is employed as the preprocessing of the datasets, we can obtain smaller datasets from the original datasets. Then the modified datasets are used as the inputs of the FNN. In this way, we can optimize the weights of FNN on small datasets obtained by feature selection instead of high-dimensional original datasets. This will make it possible to optimize the weights for FNN.

In this paper, FNN optimization problems and feature selection problems are transformed into stochastic optimization problems. We propose two optimization models to solve FNN optimization problems and feature selection optimization problems, respectively. Our idea is to optimize the weights of the neural network by combining feature selection instead of optimizing the weights only by an evolutionary algorithm. In order to effectively optimize the two models, we present a self-adaptive parameter and strategy based particle swarm optimization (SPS-PSO) algorithm. In the SPS-PSO algorithm, the parameter self-adaptive mechanism and strategy self-adaptive [22], [23] mechanism are simultaneously introduced to make it more flexible to adapt to different problems. Moreover, five candidate solution generating strategies (CSGSs) are used in SPS-PSO. Finally, by using feature selection as the preprocessing method for the datasets of FNN, we can solve the FNN optimization problems easily by EC methods, and we investigate whether similar or even better classification accuracy can be obtained. Figure 1 briefly describes the core idea in this paper.

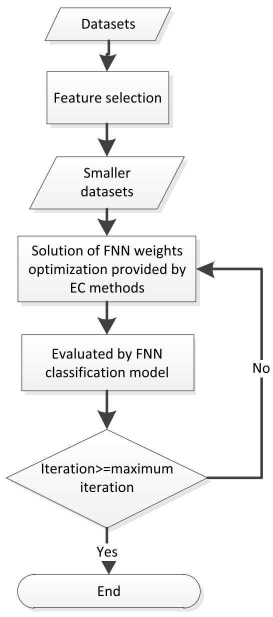

Datasets
Feature selection
Smaller datasets
Solution of FNN weights optimization provided by EC methods
Evaluated by FNN classification model
No
Iteration>=maximum iteration
Yes
End

FIGURE 1. The core idea for solving large-scale FNN optimization problems.

The experiments are performed on 8 datasets which include 3 low-dimensional datasets and 5 high-dimensional datasets. On DS 8, the dimensions of the optimization problem corresponding to the dataset reaches 845,250, which makes this a rather challenging issue. According to the proposed two optimization models, we divide the experiments into two groups as follows:

1) We use SPS-PSO to optimize the weights for FNN, and the results are compared with three other EC methods: genetic algorithm (GA) [24], particle swarm optimization (PSO) [25] and biogeography-based optimization (BBO) [26]. In this way, all the datasets without any modifications are input to the FNN weights optimization directly, including the large-scale FNN optimization models.

2) We firstly use the SPS-PSO based feature selection to generate smaller datasets from the original datasets. Then, the obtained datasets are used as the inputs of the FNN, and we use SPS-PSO and three other EC methods GA, PSO and BBO to optimize the weights of FNN again. In [27], k-nearest neighbor (KNN) [28] has been found to be effective in solving the feature selection problem when used as an evaluation function for feature selection. For saving time, KNN is used as the evaluation function in the feature selection method in this paper instead of using FNN.

At last, we compare the experimental results which include the classification accuracy and solution size of the first group and the second group, and analyze the effect of SPS-PSO based feature selection method on reducing the computational complexity of FNN optimization problems.

The rest of this paper is organized as follows: Section 2 introduces the optimization models for FNN and feature selection. Section 3 introduces the strategy and parameter self-adaptive mechanism of the SPS-PSO method. Section 4 introduces the design of the experimental groups, detailed parameter settings, and finally analyzes the results of the two experimental groups. Finally, Section 5 concludes the paper and describes the future research directions.

II. OPTIMIZATION MODELS FOR FNN AND FEATURE SELECTION

A. FNN OPTIMIZATION MODEL

In this paper, we use a simple three-layer structure for FNN [29] which contains an input layer, a hidden layer, and an output layer. The input layer gets the input values, the output layer outputs final results, and all layers between them are called hidden layers. In FNN, the nodes from different layers are connected with many weights (w).

Fig.2 shows the three-layer FNN structure. As we can see from the figure, there are weights (w) between the nodes from the different layers. In this FNN framework, the input layer has m nodes, the hidden layer has n nodes, and the output layer has k nodes, where m is the dimension of the datasets, i.e., the number of features, k is the number of class labels. Therefore, the total number of w is:

$$N_w = m * n + n * k \tag{1}$$

where N_w is the total number of w, $m*n$ represents the number of weights between the input layer z_i and the hidden layer a_j, and $n*k$ represents the number of weights between the hidden layer a_j and the output layer h_p.

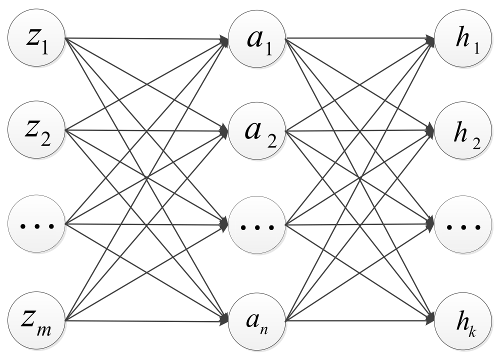


FIGURE 2. The structure of FNN.

We use z_i to represent the value of the i^{th} node of the input layer, a_j to represent the value of the j^{th} node of the hidden layer, and h_p to represent the value of the p^{th} node of the output layer. Besides, $w_{i,j}$ represents the connection weight between z_i and a_j, and $w'_{j,p}$ represents the connection weight between a_j and h_p. After inputting the values, the calculation method of a_j and h_p is shown as the following two equations:

$$a_j = f\left(\sum_{i=1}^{m} w_{i,j} * z_i + b_j\right) \tag{2}$$

$$h_p = f\left(\sum_{i=1}^{n} w'_{j,p} * a_i + b'_p\right) \tag{3}$$

where b_j and b'_p represent offset values, and the total number of them is $N_b = n+k, f(u)$ represents the activation function. The activation function is used to add nonlinear factors to solve the problems that cannot be solved by linear models. In this paper, we use sigmoid function as the activation function for FNN which is described as follows:

$$f(u) = \frac{1}{1+e^{-u}} \tag{4}$$

The main purpose of FNN weights optimization is to reduce the error between the output values and the target values of the output layers, so as to achieve higher classification accuracy. We use the mean square error (MSE) to evaluate the error between the output values and the target values. The calculation method of MSE is described as follows:

$$MSE = \sum_{p=1}^{k} (h_p^{target} - h_p^{out})^2 \tag{5}$$

where h_p^{target} represents the target value and h_p^{out} represents the output value of the node p in the output layer. A smaller value of MSE means the smaller difference between the output value and the target value, which indicates that the used weights are better.

For a dataset with s samples, the total classification error can be expressed by the average MSE of all the samples, the calculation method is described as follows:

$$fitness = \overline{MSE} = \sum_{r=1}^{s} \frac{MSE_r}{s} \tag{6}$$

where $r = 1, 2, \ldots, s$.

$\overline{MSE}$ is used as an objective function for the EC methods. In the FNN optimization process, the EC methods are used to find the optimal weights with the goal of reducing $\overline{MSE}$. In the other words, the FNN optimization process is to search the optimal solution which can make the $\overline{MSE}$ as smaller as possible.

The solution representation of the EC methods is described as follows:

$$x = [w_{i,j}, w'_{j,p}, b_j, b'_p] \tag{7}$$
$$SolutionSize = N_w + N_b \tag{8}$$

where $i = 1, 2, \ldots, m, j = 1, 2, \ldots, n, p = 1, 2, \ldots, k$, x represents an individual in the population of the EC methods.

The FNN optimization problem is to find the most suitable x to make *fitness* reach a minimum. So all kinds of EC methods can be employed to solve this problem.

B. FEATURE SELECTION OPTIMIZATION MODEL

For each feature in the feature selection problem, there are two conditions: selected and unselected. To conveniently represent feature subsets, we transform the feature selection problem into a combination optimization problem. A D-dimensional vector **B** is used to represent a solution, and the value of each dimension in **B** belongs to $\{0, 1\}$. The value in **B** means different status of feature selection: 1 represents the corresponding feature is selected and 0 is unselected.

In the initial population of SPS-PSO, we generate a D-dimensional vector **BI** with continuous values for each particle. Then a threshold $\theta \in (0, 1)$ is used to map **BI** to **B**. If the values in **BI** is bigger than θ, the corresponding dimensions in **B** will be set to 1, and 0 otherwise. To make the evolution process more efficient, we employ the mixed initialization as in [30]. In this initialization method, most particles are initialized with a small number of features, and the remaining particles are initialized with a large number of features. In [30], a method has been proven to be more efficient than the others in [30] when used as the update method of particles. During this particle update method, *pbest*/*gbest* will be updated in the following two cases:
1) The classification performance of the particle's new position is better than *pbest*/*gbest*;
2) The classification performance is the same as *pbest*/*gbest* but the number of features is smaller.

In the existing research [27], KNN has been proved to have advantages as the classifier for feature selection problems. Therefore, we use KNN as the classifier of the feature selection method in this paper for saving time instead of using FNN.

III. SELF-ADAPTIVE PARAMETER AND STRATEGY BASED PARTICLE SWARM OPTIMIZATION

We present a self-adaptive parameter and strategy based particle swarm optimization algorithm [31]. In this algorithm, strategy self-adaptive [32] and parameter self-adaptive [33] mechanism are simultaneously introduced into the traditional PSO algorithm to make it more flexible and adaptive to select the most suitable CSGS and parameter values according to different problems.

A. CANDIDATE SOLUTION GENERATION STRATEGIES (CSGSS)

The SPS-PSO algorithm uses the following five CSGSs. These CSGSs have been proved to be effective by [30], [34]–[36].

1) The CSGS in [30], as described below:

$$x_{i,d}^{t+1} = x_{i,d}^{t} + v_{i,d}^{t+1} \tag{9}$$
$$v_{i,d}^{t+1} = w * v_{i,d}^{t} + c_1 * r_1 * \left(p_{i,d} - x_{i,d}^{t}\right) + c_2 * r_2 * \left(p_{g,d} - x_{i,d}^{t}\right), \tag{10}$$

where t represents the t^{th} iteration in the evolutionary process. D is the dimensionality of the search space and $d \in D$ represents the d^{th} dimension. w is an inertia weight. On this basis, $x_{i,d}^{t}$ represents the d^{th} dimension of current particle's position. $v_{i,d}^{t} \in [-v_{\max}, v_{\max}]$ represents the velocity of the i^{th} particle in the current iteration t. c_1 and c_2 are acceleration constants. r_1 and r_2 are random values uniformly distributed in $[0, 1]$. $p_{i,d}$ and $p_{g,d}$ respectively represent the d^{th} elements of personal best solution and global best solution.

2) The position update strategy proposed in [34] is used in SPS-PSO. It is described as follows:

$$x_{i,d}^{t+1} = r_1 * x_{i,d}^{t} + r_2 * p_{g,d} + r_3 * \left(x_{a,d}^{t} - x_{b,d}^{t}\right), \tag{11}$$

where $x_{a,d}^{t}$ and $x_{b,d}^{t}$ are position vectors of two random particles. The velocity update method and other variables have the same meaning as described earlier.

3) Estimation-based velocity update strategy from [35] is used in SPS-PSO. It is described as in (12) and (13), as shown at the bottom of this page. where $N(0, 1)$ and $C(0, 1)$ represent two numbers randomly generated by the Gaussian distribution and Cauchy distribution, respectively. $mean_{i,d}^{t}$ is set to be the same as in Ref. [37]. The position update method and other variables have the same meaning as described earlier.

4) The CLPSO velocity update strategy from [36] is used in SPS-PSO. It is described as follows:

$$v_{i,d}^{t+1} = w * v_{i,d}^{t} + c_1 * r_1 * \left(pbest_{f_i(d)} - x_{i,d}^{t}\right), \tag{14}$$

$$c = \frac{(D-1)\,N(0,1)}{D} + \frac{C(0,1)}{D} \tag{12}$$

$$v_{i,d}^{t+1} = \left(mean_{i,d}^{t} - x_{id}^{t}\right) + \frac{c}{\sqrt{3}}\sqrt{\left(p_{i,d} - mean_{i,d}^{t}\right)^2 + \left(x_{i,d} - mean_{i,d}^{t}\right)^2 + \left(x_{a,d} - mean_{i,d}^{t}\right)^2}, \tag{13}$$

where $f_i = [f_i(1), f_i(2), \ldots\ldots, f_i(d)]$ defines which personal best should be used by the current particle. $pbest_{f_i(d)}$ can be the corresponding dimensionality of any particle's $pbest$ including its own $pbest$. The position update method and other variables have the same meaning as described earlier.

5) An improved CLPSO velocity update strategy called the PSO-CL-pbest strategy from [35] is used in SPS-PSO. It is described as follows:

$$v_{i,d}^{t+1} = w * v_{i,d}^{t} + 0.5 * c_1 * r_1 * \left(pbest_{f_i(d)} - x_{i,d}^{t} + p_{g,d} - x_{i,d}^{t}\right), \tag{15}$$

where the position update method and other variables have the same meaning as described earlier.

B. SELF-ADAPTIVE MECHANISM OF SPS-PSO

The strategy self-adaptive mechanism [38] and parameter self-adaptive mechanism of SPS-PSO are described as follows:

1) STRATEGY SELF-ADAPTIVE

In the strategy self-adaptive approach [37], [39], an update cycle is set for all CSGSs, the selection probability for each CSGS is fixed in each cycle. If the new solution generated by this CSGS is better than the old one, the selection probability of this CSGS will be increased, otherwise it will be reduced. First, an same initial probability $1/N_q$ is set for each CSGS, where N_q is the number of CSGSs in the strategy pool. p_q is used to represent the selection probability of the q^{th} $(q = 1, 2, 3 \ldots, N_q)$ CSGS. Then a CSGS is selected by Roulette wheel method [40] based on the selection probability. It is used to generate a new solution and will be evaluated for whether to update $pbest$ and $gbest$. Subsequently, the results is recorded by the elements $nsFlag_{i,q}$ and $nfFlag_{i,q}$ $(i = 1, 2, \ldots, ps, q = 1, 2, \ldots, N_q)$, where N_{ps} is the number of particle and N_q is the number of CSGSs, in the two matrices $nsFlag_{N_{ps}\times N_q}$ and $nfFlag_{N_{ps}\times N_q}$.

$$nsFlag = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N_{ps}\times N_q}$$
$$nfFlag = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N_{ps}\times N_q} \tag{16}$$

After finishing a generation of evolution, we calculate the sum of every column from $nsFlag_{i,q}$ and $nfFlag_{i,q}$, and record the results in two elements $S_{k,q}$ and $F_{k,q}$ $(k = 1, 2, \ldots, N_g, q = 1, 2, \ldots, N_q)$, where N_g represents the number of generations in an update cycle. $S_{k,q}$ means the number of successful evolution that the q^{th} CSGS has in the k^{th} generation. Meanwhile, the matrices $nsFlag_{N_{ps}\times N_q}$ and $nfFlag_{ps\times N_q}$ are reset according to equation (17).

$$S = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N_g\times N_q} \quad F = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N_g\times N_q} \tag{17}$$

After N_g generations, we calculate the total number of successes and failures of all CSGSs during the N_g generations, and update the selection probability of each CSGS. The calculation method is described as equation (18) and (19).

$$S_q^1 = \sum_{k=1}^{N_g} S_{k,q} \tag{18}$$

$$S_q^2 = \begin{cases} \varepsilon, & if\ S_q^1 = 0 \\ S_q^1, & otherwise \end{cases} \tag{19}$$

Then, S_q^2 is divided by their total number of times that a CSGS was selected in the last cycle as described in equation (20) and (21) to obtain the probability that each CSGS succeeds in the last N_g generations. This new probability will be used in the next evolution cycle.

$$S_q^3 = S_q^2 \Bigg/ \left(S_q^2 + \sum_{k=1}^{N_g} F_{k,q}\right) \tag{20}$$

$$p_q = S_q^3 \Bigg/ \sum_{q=1}^{N_q} S_q^3 \tag{21}$$

2) PARAMETER SELF-ADAPTIVE

In SPS-PSO, we use the values from existing literature [30], [34]–[36] for each parameter as the initial values $PVM_{N_p} = \left[P_1\ P_2 \cdots P_{N_p}\right]$, where N_p is the total number of parameters $(c_n(n = 1, 2, 3\ldots)$ in section 3.1) of all the CSGSs, P_n is the n^{th} parameter in PVM_{N_p}. At the beginning of evolving a population, all parameter values are set as equation (22).

$$PVMemory = \begin{bmatrix} P_1 & P_2 & \cdots & P_{N_p} \\ P_1 & P_2 & \cdots & P_{N_p} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}_{N_{ps}\times N_p} \tag{22}$$

If a CSGS is selected, the parameters which need to be used in this CSGS will be taken out from corresponding row and column in $PVMemory_{N_{ps}\times N_p}$. Then, a new parameter value is generated by Gaussian transformation, and we use the initial value of parameter as mean and the standard deviation is 0.3. These new parameter values will be used to generate a new particle. If the new particle performs better than the old one, these new parameter values will replace the corresponding old ones in $PVMemory_{N_{ps}\times N_p}$. After a generation of evolution, we count the means of every column in $PVMemory_{N_{ps}\times N_p}$ and record it in $TempPVM_{N_g\times N_p}$.

$$TempPVM = \begin{bmatrix} TP_1 & TP_2 & \cdots & TP_{N_p} \\ TP_1 & TP_2 & \cdots & TP_{N_p} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}_{N_g\times N_p} \tag{23}$$

Algorithm 1 Pseudo Code of the Self-Adaptive Mechanism of SPS-PSO

```
Input: number of fitness evaluations ($NFE$), current
       fitness evaluations ($cFE$), population size ($N_{ps}$),
       number of strategies ($N_q$), parameter pool
       $PVM_{N_p} = [P_1\ P_2\ \cdots\ P_{N_p}]$, selection
       probability $p_q = \frac{1}{N_q}\ (q = 1, 2, 3, \ldots, N_q)$,
       $N_g = 10$, $cur\_iter = 0$, $flag\_iter = 0$;
Output: $gbest$
1  while $cFE < NFE$ do
2      Set $PVM_{N_p}$ to
       $PVMemory = \begin{bmatrix} P_1 & P_2 & \cdots & P_{N_p} \\ P_1 & P_2 & \cdots & P_{N_p} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}_{N_{ps} \times N_p}$;
3      for $i < N_{ps}$ do
4          Select a CSGS by using $p_q$;
5          Take out $PVMemory_{i,t}$ from $PVMemory_{N_{ps} \times N_p}$,
           and generate a new $p_i$;
6          Generate a new particle by the CSGS and $p_i$;
7          Get the fitness of $x_i^{new}$;
8          if $x_i^{new}$ is better than $x_i$ then
9              $nsFlag_{i,q} = 1$;
10             $PVMemory_{i,t} = p_i$;
11             if $x_i^{new}$ is better than $pbest$ then
12                 $pbest = x_i^{new}$;
13                 if $x_i^{new}$ is better than $gbest$ then
14                     $gbest = x_i^{new}$;
15                 end
16             end
17         end
18         else
19             $nfFlag_{i,q} = 1$;
20         end
21         $cFE = cFE + 1$;
22         $x_i = x_i^{new}$;
23         $i = i + 1$;
24     end
25     $cur\_iter = cur\_iter + 1$;
26     $S_{N_g \times N_q}$ = the sum of each column in $nsFlag_{i,q}$;
27     $F_{N_g \times N_q}$ = the sum of each column in $nfFlag_{i,q}$;
28     Restore $nsFlag_{i,q}$ and $nfFlag_{i,q}$ to zero matrices;
29     $TempPVM_{N_g \times N_p}$ = the mean of the rows of
       $PVMemory_{N_{ps} \times N_p}$;
30     $PVM_{N_p} = TempPVM_{l,N_p} (l = cur_iter - flag_iter)$;
31     if $cur\_iter - flag\_iter = N_g$ then
32         Update $p_q\ (q = 1, 2, 3, \ldots, N_q)$ by $S_{N_g \times N_q}$ and
           $F_{N_g \times N_q}$;
33         Restore $S_{N_g \times N_q}$ and $F_{N_g \times N_q}$ to zero matrices;
34         $PVm_t$ = the mean of the rows of
           $TempPVM_{N_{ps} \times N_p}$;
35         Restore $TempPVMemory_{N_{ps} \times N_p}$ to zero matrix;
36         $flag\_iter$=$cur\_iter$;
37     end
38 end
```

After every N_g generations, the parameter values will be updated. The method is to get the means of every column from $TempPVM_{N_g \times N_p}$. These new parameter values then replace the old ones in $PVM_{N_p} = [P_1\ P_2\ \cdots\ P_{N_p}]$ and will be used in the next evolution cycle.

IV. EXPERIMENTS AND ANALYSIS

A. DATASETS AND BENCHMARK ALGORITHMS

In order to verify the effect of our proposed methods for optimizing the large-scale FNN, in the experiment, we have selected three low-dimensional datasets and five high-dimensional datasets. Although the datasets are not very high in dimension, when it is used to validate the large-scale FNN problems, the solution to its corresponding optimization problem is very large. Therefore, we think it is enough to describe our new idea with the employed datasets in the manuscript. The datasets used in the experiments are chosen from the University of California Irvine (UCI) Machine Learning Repository [41] and Causality Workbench [42]. Among them, 7 datasets are derived from UCI, except DS 4 from Causality Workbench.

TABLE 1. Information of datasets.

ID	Datasets	NoE	NoF	NoC
DS1	diabetes	768	8	2
DS2	wdbc	569	30	2
DS3	ConnectionistBench	208	60	2
DS4	musk	476	166	2
DS5	LVST	126	310	2
DS6	madelon	2000	500	2
DS7	isolet5	120	617	2
DS8	MultipleFeaturesDigit	400	649	2

In Table 1, "DS n" represents the n^{th} dataset, NoE means the number of examples, NoF means the number of features (dimensions), and NoC is the number of classes. Besides, we divide all the datasets into training sets and test sets. Among them, the training sets account for 70% while the remaining 30% are used as test sets.

In the experiments, we select some EC methods such as genetic algorithm (GA) [24], particle swarm optimization (PSO) [25] and biogeography-based optimization (BBO) [26] as the comparison algorithms. The experiments are divided into two groups:

1) We use SPS-PSO and three comparison algorithms to optimize the weights of FNN on the original datasets without any modification. Then compare the results of them.

2) Firstly, SPS-PSO based feature selection method is used to generate smaller datasets from the original datasets. Secondly, these datasets are used as the inputs of FNN, and we use the EC methods to optimize the weights of FNN again. Finally, we compare the obtained results with that of the first experimental group.

For convenience, we use "FS-EC" to indicate the results with feature selection and "EC" to indicate the results without feature selection.

TABLE 2. Parameter settings of experiments.

Experimental Parameters				
NFE_{FS}	NFE_{FNN}	N_g	N_{ps}	Re
100000	300000	10	100	30

B. PARAMETER SETTINGS

The detailed experimental parameter settings are given in Table 2, where NFE_{FS} represents the maximum number of fitness evaluation in feature selection, NFE_{FNN} represents the maximum number of fitness evaluation in FNN, N_g is the number of the leaning period which is the number of generations between two updated cycle. For SPS-PSO, we set 0.5 as the initial value for the parameters that need to be involved in the adaptive mechanism in all CSGSs.

To compare the performance of different algorithms as fair as possible, we use the same NFE_{FNN} for different algorithms. n order to get more reliable results, all the experiments are repeated for Re times.

C. RESULTS AND ANALYSIS

The results of the two experimental groups are shown in Tables 3-6. In these tables, ''mean'' represents the average of the accuracy of the final solutions obtained by each algorithm, ''std'' represents the standard deviations, ''size'' is the average solution size of inputted datasets in FNN. ''+'',''−'' and ''='' represent whether exist the significant difference between an algorithm and others.

1) RESULTS OF DIFFERENT EC METHODS FOR TRAINING FNN

Tables 3-4 show the results on training and test sets between the SPS-PSO and the other three EC methods on optimizing the weights for FNN. Fig.2 are the convergences of $\overline{MSE}$ in the process of optimizing the FNN model by four EC methods.

From the results, we can see that SPS-PSO gets the highest accuracy on DS 1,2,5,7,8. BBO gets the best accuracy on DS 2,3,4,7,8. PSO performs best on DS 6 and GA performs the same as BBO and SPS-PSO on DS 7, 8. Obviously, BBO

TABLE 3. Classification of EC methods on training sets.

Datasets	GA		PSO		BBO		SPS-PSO	
	mean	std	mean	std	mean	std	mean	std
DS1	8.47E-01	6.80E-03	8.34E-01	4.40E-03	8.59E-01	1.90E-03	**8.76E-01**	7.80E-03
	+		+		+			
DS2	9.95E-01	4.10E-03	9.93E-01	3.10E-03	**9.97E-01**	3.90E-03	**9.97E-01**	3.50E-03
	+		+		=			
DS3	9.74E-01	3.30E-02	8.79E-01	2.58E-02	**1.00E+00**	0.00E+00	9.98E-01	5.20E-03
	+		+		-			
DS4	8.60E-01	5.35E-02	7.80E-01	2.04E-02	**9.83E-01**	9.70E-03	9.44E-01	1.77E-02
	+		+		-			
DS5	8.17E-01	1.49E-01	8.63E-01	1.86E-02	9.68E-01	2.53E-02	**9.69E-01**	1.50E-03
	+		+		=			
DS6	5.51E-01	1.09E-02	**5.59E-01**	0.00E+00	5.42E-01	1.33E-02	5.44E-01	0.00E+00
	-		-		=			
DS7	**1.00E+00**	0.00E+00	9.84E-01	1.46E-02	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00
	=		+		=			
DS8	**1.00E+00**	0.00E+00	9.96E-01	5.30E-03	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00
	=		+		+			

TABLE 4. Classification of EC methods on test sets.

Datasets	GA		PSO		BBO		SPS-PSO	
	mean	std	mean	std	mean	std	mean	std
DS1	**8.50E-01**	6.60E-03	8.31E-01	1.06E-02	8.48E-01	2.30E-03	**8.50E-01**	6.20E-03
	=		+		+			
DS2	9.86E-01	1.70E-03	9.86E-01	6.70E-03	9.93E-01	3.50E-03	**9.94E-01**	7.00E-04
	+		+		=			
DS3	8.26E-01	5.44E-02	7.60E-01	3.29E-02	8.84E-01	1.38E-02	**8.88E-01**	3.70E-02
	+		+		=			
DS4	7.80E-01	5.09E-02	7.30E-01	2.30E-02	**9.19E-01**	2.32E-02	8.54E-01	1.57E-02
	+		+		-			
DS5	7.39E-01	9.64E-02	7.07E-01	6.11E-02	8.38E-01	4.08E-02	**8.42E-01**	8.41E-02
	+		+		=			
DS6	5.09E-01	5.60E-03	5.19E-01	0.00E+00	**5.41E-01**	1.13E-02	5.02E-01	0.00E+00
	-		-		-			
DS7	**1.00E+00**	0.00E+00	9.34E-01	4.01E-02	9.73E-01	1.78E-02	9.88E-01	5.90E-03
	-		+		+			
DS8	9.85E-01	1.32E-02	9.86E-01	1.30E-02	9.93E-01	4.40E-03	**9.97E-01**	4.80E-03
	+		+		+			

TABLE 5. Classification of EC methods and FS-EC methods on training sets.

Datasets	GA			FS-GA			PSO			FS-PSO		
	mean	std	size	mean	std	size	mean	std	size	mean	std	size
DS1	**8.47E-01** -	6.80E-03	171	8.40E-01	5.00E-03	66	**8.34E-01** -	4.40E-03	171	8.20E-01	2.23E-02	72
DS2	9.95E-01 =	4.10E-03	1953	**9.96E-01**	2.70E-03	351	**9.93E-01** -	3.10E-03	1953	9.90E-01	3.20E-03	319
DS3	9.74E-01 +	3.30E-02	7503	**9.84E-01**	1.24E-02	780	8.79E-01 +	2.58E-02	7503	**8.95E-01**	3.25E-02	480
DS4	8.60E-01 +	5.35E-02	55945	**9.47E-01**	1.95E-02	5565	7.80E-01 +	2.04E-02	55945	**8.21E-01**	4.69E-02	5025
DS5	8.17E-01 +	1.49E-01	193753	**9.47E-01**	3.24E-02	10841	8.63E-01 =	1.86E-02	193753	**8.74E-01**	4.30E-02	9591
DS6	5.51E-01 +	1.09E-02	502503	**6.74E-01**	1.95E-02	14070	5.59E-01 +	0.00E+00	502503	**5.82E-01**	2.71E-02	12960
DS7	**1.00E+00** =	0.00E+00	764466	**1.00E+00**	0.00E+00	190	9.84E-01 +	1.46E-02	764466	**1.00E+00**	0.00E+00	226
DS8	**1.00E+00** =	0.00E+00	845650	**1.00E+00**	0.00E+00	2129	9.96E-01 =	5.30E-03	845650	**9.98E-01**	2.30E-03	2380

TABLE 6. Classification of EC methods and FS-EC methods on training sets.

Datasets	BBO			FS-BBO			SPS-PSO			FS-SPS-PSO		
	mean	std	size	mean	std	size	mean	std	size	mean	std	size
DS1	**8.59E-01** -	1.90E-03	171	8.37E-01	1.12E-02	72	**8.76E-01** -	7.80E-03	171	8.46E-01	6.60E-03	78
DS2	**9.97E-01** =	3.90E-03	1953	9.96E-01	2.00E-03	294	**9.97E-01** +	3.50E-03	1953	9.92E-01	3.80E-03	385
DS3	**1.00E+00** -	0.00E+00	7503	9.88E-01	6.10E-03	612	**9.98E-01** -	5.20E-03	7503	9.66E-01	1.24E-02	639
DS4	9.83E-01 +	9.70E-03	55945	**9.67E-01**	4.00E-03	5539	9.44E-01 +	1.77E-02	55945	**9.69E-01**	1.62E-02	4512
DS5	9.68E-01 +	2.53E-02	193753	**9.73E-01**	1.96E-02	9800	9.69E-01 -	1.50E-03	193753	**9.79E-01**	2.38E-02	11325
DS6	7.42E-01 +	1.33E-02	502503	**7.83E-01**	3.69E-02	13041	5.44E-01 +	0.00E+00	502503	**5.94E-01**	1.14E-02	12680
DS7	**1.00E+00** =	0.00E+00	764466	**1.00E+00**	0.00E+00	171	**1.00E+00** =	0.00E+00	764466	**1.00E+00**	0.00E+00	185
DS8	**1.00E+00** =	0.00E+00	845650	**1.00E+00**	0.00E+00	2538	**1.00E+00** =	0.00E+00	845650	**1.00E+00**	0.00E+00	2244

and SPS-PSO perform much better than GA and PSO. BBO and SPS-PSO are better on 5 datasets, respectively. From the significant difference, SPS-PSO is better than the GA and PSO on 7 datasets. Compared to the method BBO, SPS-PSO is not as good as BBO on DS 3,4 while is superior to BBO on the other 6 datasets. For the test sets, SPS-PSO performs best on DS 1,2,3,5,8. At the same time, GA is better on DS 1,7 and BBO is better on DS 4,6. Consider the classification accuracy, SPS-PSO has the highest accuracy on 5 datasets. Combined with significant differences, we can find that the performance of SPS-PSO is better than or equal to the other 3 EC methods on the other 6 datasets except for slightly weaker on DS 4,6.

From the figure we can see that the convergence speed and the final convergence result of GA are worse than the other three methods. The PSO converges faster, but the final result is not good enough while only better on the DS6. BBO and SPS-PSO perform better, and SPS-PSO has advantages in four datasets DS1, 2, 3, and 5. At the same time, the two performed similarly on DS7 and 8, and the results are all good. From the convergence curve, it can be found that SPS-PSO has some advantages over the other three methods in terms of convergence speed and final result.

Combined with Table 1, 2 and Fig. 2, we can conclude that SPS-PSO has advantages over the other 3 EC methods in optimizing the weights for FNN.

2) RESULTS OF FS-EC AND EC EXPERIMENTAL GROUPS

Tables 5-8 show the results of the second group of experiments.

It can be seen from Tables 5,6, after the SPS-PSO based feature selection method, the solution sizes of the EC methods for FNN weights optimization are greatly reduced, especially on high-dimensional datasets. For example, on DS 7, the size of the solution is reduced from the original 764, 466 to only 190. Compared with the original solution, such a small solution can greatly reduce the difficulty of searching the solution space and improve the computational efficiency greatly. It becomes possible to perform FNN weights optimization on high-dimensional datasets that were impossible before. More important, it is gratifying that the greatly

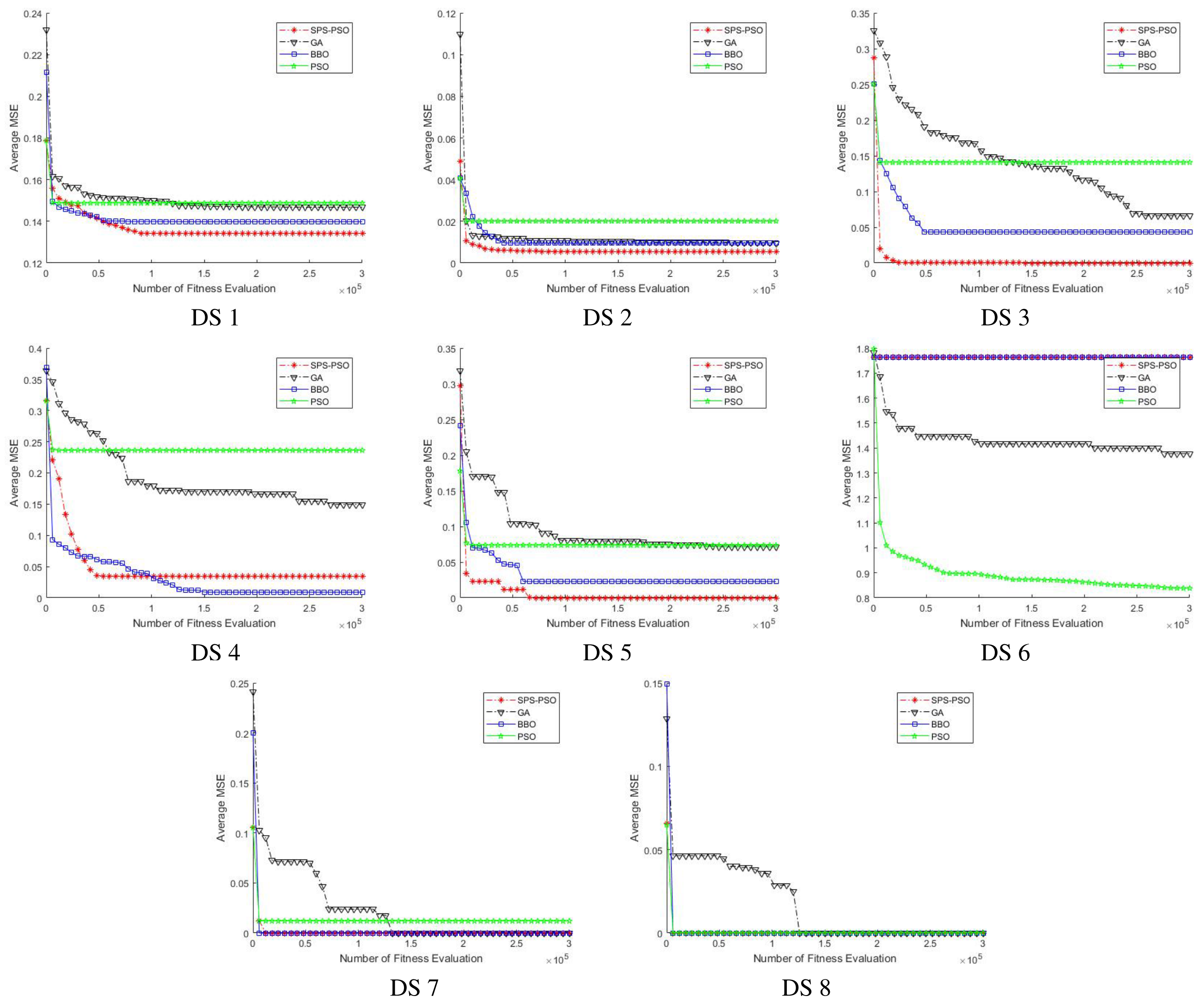


FIGURE 3. Convergence curves of $\overline{MSE}$ for different algorithms on training sets.

reduced solution does not decrease the final classification accuracy, but makes the results even better. For example, comparing the results of GA and FS-GA on training sets, we can find that FS-GA is better than or equal to GA on 7 datasets except for DS 1. FS-PSO is better than PSO on DS 3,4,5,6,7,8 except for DS 1,2. The performance of FS-BBO and FS-SPS-PSO are similar, both worse than BBO or SPS-PSO on DS 1,2,3, but better on the other 5 datasets. The results on the test set are also similar to the results on the training set. Except for the results on DS 1,2, and 3, FS-GA and FS-PSO performed better on the other 5 datasets than GA and PSO. For FS-BBO and FS-SPS-PSO, on DS1,2,3 and 5, it is slightly worse than BBO and SPS-PSO. On the low-dimensional datasets DS 1,2,3, we can find that FS-EC group performs worse than EC group. But as the dimension of the datasets increase, the advantages of FS-EC group begin to appear. On high-dimensional datasets DS 4-8, the solution sizes increase fast as the datasets become bigger and bigger. In this case, the SPS-PSO based feature selection method effectively reduces the sharply increased features with the increasing of the dimension of datasets. By feature selection, it can avoid searching for the optimal solution in a huge solution space, so as to reduce the possibility of falling into local optima. The experimental results also illustrate this point. Begin with DS 4, the dimensions of the datasets reach to 166, and the solution size is 55945. The FS-EC group is better than the EC group on the following 5 datasets DS 4-8.

In summary, by comparing the original EC group and FS-EC group, we find that the SPS-PSO based feature selection method can reduce the computational complexity of the classification by reducing the solution for the FNN weights optimization problems. It also guarantees excellent classification accuracy especially on high-dimensional datasets. Moreover, as a preprocessing method for datasets of FNN,

TABLE 7. Classification of EC methods and FS-EC methods on test sets.

Datasets	GA			FS-GA			PSO			FS-PSO		
	mean	std	size	mean	std	size	mean	std	size	mean	std	size
DS1	**8.52E-01** -	6.60E-03	171	8.40E-01	7.60E-03	66	**8.31E-01** -	1.06E-02	171	8.24E-01	9.20E-03	72
DS2	**9.96E-01** -	1.70E-03	1953	9.89E-01	6.10E-03	351	**9.86E-01** -	6.70E-03	1953	9.82E-01	7.90E-03	319
DS3	**8.26E-01** -	5.44E-02	7503	7.73E-01	8.09E-02	780	**7.60E-01** -	3.29E-02	7503	7.06E-01	5.19E-02	480
DS4	7.80E-01 +	5.09E-02	55945	**8.86E-01**	2.45E-02	5565	7.30E-01 +	2.30E-02	55945	**7.60E-01**	4.01E-02	5025
DS5	7.39E-01 +	9.64E-02	193753	**7.96E-01**	9.66E-02	10841	7.07E-01 +	6.11E-02	193753	**7.46E-01**	6.33E-02	9591
DS6	5.09E-01 +	5.60E-03	502503	**5.77E-01**	3.17E-02	14070	5.19E-01 +	0.00E+00	502503	**5.50E-01**	2.13E-02	12960
DS7	**1.00E+00** =	0.00E+00	764466	**1.00E+00**	0.00E+00	190	9.34E-01 +	4.01E-02	764466	**9.95E-01**	7.40E-03	226
DS8	9.85E-01 +	1.32E-02	845650	**9.99E-01**	1.20E-03	2129	9.86E-01 =	1.30E-02	845650	**9.91E-01**	5.00E-03	2380

TABLE 8. Classification of EC methods and FS-EC methods on test sets.

Datasets	BBO			FS-BBO			SPS-PSO			FS-SPS-PSO		
	mean	std	size	mean	std	size	mean	std	size	mean	std	size
DS1	**8.48E-01** -	2.30E-03	171	8.40E-01	5.00E-03	72	**8.50E-01** -	6.20E-03	171	8.45E-01	2.60E-03	78
DS2	**9.93E-01** -	3.50E-03	1953	9.90E-01	3.90E-03	294	**9.94E-01** -	7.00E-04	1953	9.93E-01	2.60E-03	385
DS3	**8.84E-01** -	1.38E-02	7503	8.17E-01	4.92E-02	612	**8.88E-01** -	3.70E-02	7503	7.36E-01	3.03E-02	639
DS4	9.19E-01 +	2.32E-02	55945	**9.36E-01**	9.80E-03	5539	8.54E-01 +	1.57E-02	55945	**9.41E-01**	1.92E-02	4512
DS5	**8.38E-01** -	4.08E-02	193753	8.13E-01	4.05E-02	9800	**8.42E-01** =	8.41E-02	193753	8.37E-01	3.74E-02	11325
DS6	5.41E-01 +	1.13E-02	502503	**5.63E-01**	2.05E-02	13041	5.02E-01 +	0.00E+00	502503	**5.54E-01**	2.15E-02	12680
DS7	9.73E-01 +	1.78E-02	764466	**1.00E+00**	1.10E-03	171	9.88E-01 +	5.90E-03	764466	**1.00E+00**	0.00E+00	185
DS8	9.93E-01 +	4.40E-03	845650	**1.00E+00**	4.00E-04	2538	9.97E-01 +	4.80E-03	845650	**9.99E-01**	1.50E-03	2244

the SPS-PSO based feature selection method makes it possible to optimize the weights of FNN with high-dimensional datasets by EC methods.

V. CONCLUSION

This paper transforms the large-scale FNN optimization problems and feature selection optimization problems into stochastic optimization problems which can be solved by EC methods. Then we present a parameter and strategy self-adaptive particle swarm optimization algorithm and use it to solve the large-scale FNN optimization problem. Moreover, in order to reduce the computational complexity of large-scale FNN optimization problems, we use the SPS-PSO based feature selection method as the preprocessing method for datasets of FNN.

The experimental results show that SPS-PSO has advantages in optimizing the weights for FNN compared with the other three EC methods. Moreover, when the SPS-PSO based feature selection method is applied to preprocess the inputs of FNN, the solution size can be greatly reduced. The results show that the SPS-PSO based feature selection makes it possible to optimize the large-scale FNN weights optimization problems. In the future work, we will further improve performance of the SPS-PSO algorithm on optimizing the weights of the FNN.

REFERENCES

[1] M. Y. Rafiq, G. Bugmann, and D. J. Easterbrook, "Neural network design for engineering applications," *Comput. Struct.*, vol. 79, no. 17, pp. 1541–1552, 2001.

[2] W. Cao, X. Wang, Z. Ming, and J. Gao, "A review on neural networks with random weights," *Neurocomputing*, vol. 275, pp. 278–287, Jan. 2018.

[3] X. Wang, X. Li, and V. C. M. Leung, "Artificial intelligence-based techniques for emerging heterogeneous network: State of the arts, opportunities, and challenges," *IEEE Access*, vol. 3, pp. 1379–1391, 2015.

[4] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," *Comput. Sci.*, vol. 49, pp. 907–940, Jun. 2016.

[5] S. C. Kak, "Feedback neural networks: New characteristics and a generalization," *Circuits, Syst. Signal Process.*, vol. 12, no. 2, pp. 263–278, 1993.

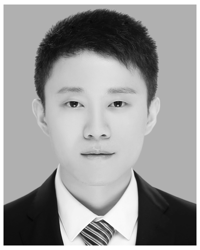
[6] F. Zhao, Y. Zhang, and Q. Wang, "Multi-slot spectrum auction in heterogeneous networks based on deep feedforward network," *IEEE Access*, vol. 6, pp. 45113–45119, 2018.

[7] T.-N. Le, P. T. Bao, and H. T. Huynh, "Liver tumor segmentation from mr images using 3D fast marching algorithm and single hidden layer feedforward neural network," *Biomed. Res. Int.*, vol. 2016, pp. 1–8, Jul. 2016.

[8] X. Yang, H. Sun, X. Sun, M. Yan, Z. Guo, and K. Fu, "Position detection and direction prediction for arbitrary-oriented ships via multitask rotation region convolutional neural network," *IEEE Access*, vol. 6, pp. 50839–50849, 2018.

[9] V. Karri and T. Kiatcharoenpol, "Prediction of internal surface roughness in drilling using three feedforward neural networks—A comparison," in *Proc. 9th Int. Conf. Neural Inf.*, vol. 4, Nov. 2002, pp. 1890–1894.
[10] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[11] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Inf. Sci.*, vol. 378, pp. 484–497, Feb. 2017.
[12] Y. Xue, B. P. Zhao, T. Ma, and A. X. Liu, "An evolutionary classification method based on fireworks algorithm," *Int. J. Bio-Inspired Comput.*, vol. 11, no. 3, pp. 149–158, 2018.
[13] Z. H. Zhou, Z. Q. Chen, and S. F. Chen, "Improving fault-tolerance of feedforward neural networks with genetic algorithms," *J. Comput. Res. Develop.*, vol. 38, no. 9, pp. 1061–1065, Sep. 2001.
[14] A. Radi and R. Poli, *Discovering Efficient Learning Rules for Feedforward Neural Networks Using Genetic Programming*. Berlin, Germany, 2003, pp. 133–159.
[15] 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.*, Apr. 2003, pp. 110–117.
[16] H. K. Lam, S. H. Ling, F. H. F. Leung, and P. K. S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," in *Proc. 27th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, vol. 1, Nov./Dec. 2001, pp. 25–30.
[17] Y. Zhang, P. Phillips, S. Wang, G. Ji, J. Yang, and J. Wu, "Fruit classification by biogeography-based optimization and feedforward neural network," *Expert Syst.*, vol. 33, no. 3, pp. 239–253, 2016.
[18] V. K. Bohat and K. V. Arya, "An effective gbest-guided gravitational search algorithm for real-parameter optimization and its application in training of feedforward neural networks," *Knowl.-Based Syst.*, vol. 143, pp. 192–207, Mar. 2018.
[19] W. Zhou, C. Wu, Y. Yi, and G. Luo, "Structure preserving non-negative feature self-representation for unsupervised feature selection," *IEEE Access*, vol. 5, pp. 8792–8803, 2017.
[20] S. Adams, A. P. Beling, and R. Cogill, "Feature selection for hidden Markov models and hidden semi-Markov models," *IEEE Access*, vol. 4, pp. 1642–1657, 2016.
[21] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
[22] Y. Xue, J. Jiang, T. Ma, J. Liu, and W. Pang, "A self-adaptive artificial bee colony algorithm with symmetry initialization," *J. Internet Technol.*, vol. 19, no. 5, pp. 1347–1362, 2018.
[23] Y. Xue, J. Jiang, B. Zhao, and T. Ma, "A self-adaptive artificial bee colony algorithm based on global best for global optimization," *Soft Comput.*, vol. 22, no. 9, pp. 2935–2952, 2018.
[24] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proc. Int. Joint Conf. Artif. Intell.*, 1989, pp. 762–767.
[25] R. Mendes, P. Cortez, M. Rocha, and J. Neves, "Particle swarms for feedforward neural network training," in *Proc. Int. Joint Conf. Neural Netw.*, May 2002, pp. 1895–1899.
[26] A. Rodan, H. Faris, and J. Alqatawna, "Optimizing feedforward neural networks using biogeography based optimization for e-mail spam identification," *Int. J. Commun., Netw. Syst. Sci.*, vol. 9, no. 1, pp. 19–28, 2016.
[27] Y. Xue, W. Jia, X. Zhao, and W. Pang, "An evolutionary computation based feature selection method for intrusion detection," *Secur. Commun. Netw.*, vol. 2018, pp. 1–10, Oct. 2018.
[28] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.
[29] R. Kothari and K. Agyepong, "Self-regulation of model order in feedforward neural networks," in *Proc. Int. Conf. Neural Netw.*, vol. 3, Jun. 1997, pp. 1966–1971.
[30] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms," *Appl. Soft Comput.*, vol. 18, no. 4, pp. 261–276, May 2014.
[31] M. Juneja and S. K. Nagar, "Particle swarm optimization algorithm and its parameters: A review," in *Proc. Int. Conf. Control, Comput., Commun. Mater. (ICCCCM)*, Oct. 2016, pp. 1–5.
[32] Y. Xue, B. Zhao, T. Ma, and W. Pang, "A self-adaptive fireworks algorithm for classification problems," *IEEE Access*, vol. 6, pp. 44406–44416, 2018.
[33] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[34] H. Wang, "Diversity enhanced particle swarm optimization with neighborhood search," *Inf. Sci.*, vol. 223, no. 2, pp. 119–135, Feb. 2013.
[35] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Inf. Sci.*, vol. 181, no. 20, pp. 4515–4538, 2011.
[36] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
[37] Y. Xue, S. Zhong, Y. Zhuang, and B. Xu, "An ensemble algorithm with self-adaptive learning techniques for high-dimensional numerical optimization," *Appl. Math. Comput.*, vol. 231, pp. 329–346, Mar. 2014.
[38] Y. Xue, Y. Zhuang, T. Ni, J. Ouyang, and Z. Wang, "Enhanced self-adaptive evolutionary algorithm for numerical optimization," *J. Syst. Eng. Electron.*, vol. 23, no. 6, pp. 921–928, Dec. 2012.
[39] Y. Xue, Y. Zhuang, T. Ni, S. Ni, and X. Wen, "Self-adaptive learning based discrete differential evolution algorithm for solving CJWTA problem," *J. Syst. Eng. Electron.*, vol. 25, no. 1, pp. 59–68, Feb. 2014.
[40] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 3–14, Jan. 1994.
[41] K. Bache and M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml/index.php
[42] Causality Workbench Team. *Causality Workbench Data-Repository*. Accessed: Jul. 2018. [Online]. Available: http://www.causality.inf.ethz.ch

YU XUE received the Ph.D. degree from the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China, in 2013. He was a Visiting Scholar with the School of Engineering and Computer Science, Victoria University of Wellington. He is currently an Associate Professor with the School of Computer and Software, Nanjing University of Information Science and Technology. He is also a Visiting Scholar with the Department of Computer Science and Engineering, Michigan State University, East Lansing, USA. His research interests include evolutionary computation, machine learning, and data mining. He is a member of ACM and CCF.



TAO TANG received the bachelor's degree from the School of Computer and Software, Nanjing University of Information Science and Technology, China, where he is currently pursuing the master's degree with the School of Computer and Software. His research interests include evolutionary algorithms, self-adaptive search, evolutionary multi-objective optimization, feature selection, and machine learning.

ALEX X. LIU is currently a Professor with the Department of Computer Science and Engineering, Michigan State University. His research interests include networking, security, and dependable systems. He was a recipient of the IEEE and IFIP William C. Carter Award, in 2004, and the U.S. National Science Foundation (NSF) Career Award, in 2009. He is an Associate Editor of the IEEE/ACM Transactions on Networking, the IEEE Transactions on Dependable and Secure Computing, and the IEEE Transactions on Mobile Computing.