Intelligent Neural Network with Parallel Salp Swarm Algorithm for Power Load Prediction

Jin-Liang Zhou[1], Shu-Chuan Chu[1,2], Ai-Qing Tian[1], Yan-Jun Peng[1], Jeng-Shyang Pan[1,3*]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, China
[2] College of Science and Engineering, Flinders University, Australia
[3] Department of Information Management, Chaoyang University of Technology, Taiwan
zhoujinliang@sdust.edu.cn, scchu0803@gmail.com, stones12138@163.com, pengyanjuncn@163.com, jspan@cc.kuas.edu.tw

Abstract

The neural network runs slowly and lacks accuracy in power load forecasting, so it is optimized using meta-heuristic algorithm. Salp Swarm Algorithm (SSA) is a novel meta-heuristic algorithm that simulates the salp foraging process. In this paper, a parallel salp swarm algorithm (PSSA) is proposed to improve the performance of SSA. It not only improves local development capabilities, but also accelerates global exploration. Through 23 test functions, PSSA performs better than other algorithms and can effectively explore the whole search space. Finally, PSSA is used to optimize the weights as well as the thresholds of the neural network. Using the optimized neural network to predict the power load in a certain region, the results show that PSSA can better optimize the neural network and increase its prediction accuracy.

Keywords: Power load forecast, Salp swarm algorithm, Artificial neural network, Particle swarm algorithm

1 Introduction

Power load forecasting is an important component of electric system engineering, and it is extremely important for the stable operation and long-term planning of the power system [1-2]. Load forecasting is based on past power demand, weather conditions, temperature, humidity, and other related factors, using mathematical models to predict possible future electricity consumption. The purpose of load forecasting is to simulate future electricity demand as closely as possible, to equalize supply and demand, and to mitigate supply-demand conflicts [3]. On the one hand, too high a forecast will generate too much electricity, resulting in wasted resources, and too low a forecast will result in a short circuit, which will lead to a massive blackout or power system collapse if the low-frequency load shedding unit fails to reduce customer load at this time. On the other hand load forecasting is crucial for planning the construction of future power grids; too high forecasts will lead to the construction of useless facilities and increased costs, while too low forecasts will lead to inadequate long-term planning and affect the development of the power system [4].

Depending on the length of the forecast time span, load forecasting can be divided into four types. The first type is ultra-short-term load forecasting, which is used to forecast the load for a few minutes or hours. This provides support for monitoring the operating conditions of power equipment. The second type is Short Term Load Forecasting (STLF), which is used to forecast load for hours, days and weeks. This provides a reference for power scheduling, start-up and shutdown of generating units, etc. The third type is the Medium Term Load Forecast (MTLF), which is used to forecast load for several weeks or months [5]. Its accuracy has a significant impact on the economics of the electric utility [6-7]. Electric utilities make many decisions based on these forecasts. These decisions include economic dispatch of generation capacity, system safety assessment, equipment maintenance, and fuel procurement planning. The fourth type is long-term load forecasting (LTLF), which is used to forecast load for months or even years into the future. It is mainly used for power system planning.

Currently, forecasting techniques can be broadly classified into two categories. The first category is statistical models, including time series methods, trend extrapolation methods, and regression analysis. Such methods are based on a series of mathematical formulas and make forecasts based on previous loads. The time series method is a statistical model that takes advantage of the inertial characteristics and temporal continuity of electric load changes, and determines the basic characteristics and change patterns of load by analyzing and processing the time series of historical data, so as to forecast future load. Time series forecasting methods can be divided into two categories: deterministic and stochastic. Deterministic time series are used as model residuals to estimate the size of the forecast interval. The stochastic time series forecasting model can be regarded as a linear filter. According to the characteristics of the linear filter, the time series can be classified into several types of models: auto regression (AR), moving average (MA), autoregressive moving average (ARMA), cumulative autoregressive-moving average (ARIMA), and transfer function (TF). These models are characterized by good accuracy under classical conditions, but poorer results when outliers appear [8]. The second class is intelligent forecasting methods represented by neural networks (NN) [9-12], support vector machines (SVM) [2，13-14], and long short-term memory (LSTM) [15-16]. Artificial neural networks (ANN), as an intelligent technology, have been applied in various fields such as information processing, pattern recognition, feature selection, and price prediction. There are various types of artificial neural

*Corresponding Author: Jeng-Shyang Pan; E-mail: jspan@cc.kuas.edu.tw
DOI: 10.53106/160792642022072304001

networks, such as backpropagation neural networks (BPNN), which can approximate any nonlinear function very accurately. By learning from historical data, a network that approximates the relationship between the input vectors and the corresponding output vectors is obtained, which in turn can effectively predict new data. In terms of load prediction, BP neural networks have many advantages, such as powerful nonlinear mapping capability, self-learning ability, and generalization capability. However, it also has obvious disadvantages, such as long-running time, slow convergence, and ease to fall into local optimality. Due to the limitations of neural networks themselves, intelligent computing is introduced to optimize neural networks.

Intelligent computing was developed from general-purpose computing and has become one of the important approaches in the field of computer science. For the past few years, intelligent computing technology has been successful in the military, financial engineering, route plan, pattern recognition, wireless sensor network, computer-aided medical diagnosis, and other fields [17-19]. Intelligent computing is also known as computational intelligence, and computational intelligence includes three types of neural computing, fuzzy computing and evolutionary computing. Among them, evolutionary computation includes Particle Swarm Optimization (PSO) [20-25], Grey Wolf Optimizer (GWO) [26-28], Ant Colony Optimization (ACO) [29-31], Bat Algorithm (BA) [32-34], Cat Swarm Optimization (CSO) [35-37], QUasi-Affine TRansformation Evolutionary (QUATRE) [38-40], Flower Pollination Algorithm (FPA) [41], Genetic Algorithm (GA) [42-43], Cuckoo Search Algorithm (CS) [44-47], Differential Evolution (DE) [48-50], Pigeon-Inspired Algorithm (PIO) [51], and Multi-Verse Optimizer (MVO) [52-53]. Evolutionary computing algorithms mainly include exploration and development. Exploration means finding candidate solutions, and development means finding optimal solutions based on better candidate solutions. Each of the above-proposed algorithms has its advantages and disadvantages, for example, PSO has the advantages of fast search speed and easy implementation, however, it tends to fall into local optima, resulting in low convergence accuracy. GA does not include a storage function, so the information related to the best particle is not stored during the optimization process. Some other optimization algorithms, such as GWO and FPA have been used to optimize the weights of neural networks. Salp swarm algorithm (SSA) is a recently proposed meta-heuristic algorithm that modelling the foraging behavior of salp in the deep sea [54]. SSA is easy to implement with high robustness and simple parameters, so it is widely used in feature selection [55], control parameter optimization [56], load prediction, and other fields. However, SSA still suffers from the shortcomings of falling into local optimal and evolutionary stagnation. Therefore, a parallel salp swarm algorithm (PSSA) is proposed in this study. Finally, to improve the prediction accuracy of the neural network, the PSSA algorithm is used to optimize the structure as well as the parameters of the neural network.

The remaining components of this paper are as follows: Section 2 introduces some related research work. Section 3 introduces the salp swarm algorithm. Section 4 uses parallel strategies to improve salp swarm algorithm. Section 5 uses 23 test functions to verify the performance of PSSA. Section 6 is to use PSSA to optimize the neural network and perform power load forecasting. Section 7 concludes the works of this paper, raises some questions and plans future work.

2 Related Works

The data obtained in engineering practice often have missing and duplicate values. These dirty data can significantly affect the effectiveness of the prediction work. Therefore Subsection 2.1 introduces several methods for processing the data. An artificial neural network is a mathematical model that simulates the human nervous system to process complex information. It has good adaptive and self-learning capabilities and can be used to solve nonlinear problems. Subsection 2.2 introduces the way neural networks work.

2.1 Data Pre-processing

A large amount of real and accurate historical load data is the basis for power load forecasting. However, dirty data is inevitable, such as data loss and data anomalies from various causes. These data play a significant disruptive role in the network training and forecasting process. They will lead to training errors beyond the expected range, thus weakening the prediction accuracy. Therefore, historical data must be processed to ensure the smoothness and order of magnitude consistency of the load series. Ultimately, data pre-processing will improve the accuracy of load forecasting.

2.1.1 Missing Data Processing

The missing information can affect the correct operation of the neural network. Therefore, a portion of the data with the highest correlation to the missing values can be selected based on correlation analysis. Finally, the weighted average of these data is used as the complementary value. In the data set used in this paper, a total of 0.55% of the data is lost.

$$l_d = w_1 l_{d1} + w_2 l_{d2} + \ldots + w_m l_{dm}. \quad \textbf{(1)}$$

Equation 1 shows how to calculate the missing values. Where l_d represents the missing data, $l_{di}(i = 1, 2, \ldots, m)$ represents the *ith* data related to the missing data, and w indicates the correlation between the relevant data and the missing data.

2.1.2 Abnormal Data Processing

An outlier is a set of measurements that deviate from the mean by more than a few standard deviations. Outliers can disrupt the regularity of the entire data series, which in turn affects the accuracy of the prediction. Therefore, it is necessary to correct anomalous data. In general, the load data on a particular day of the year is roughly the same, with obvious seasonal characteristics. In other words, these data are roughly cyclical in nature. Therefore, the deviation rate between the load value of that day and the average value can be calculated, and the data with a high deviation rate can be corrected. Based on the distribution of deviation rates for this dataset, this paper set deviation rate to *1.4* for spring and autumn and *1.6* for summer and winter.

2.1.3 The Normalization Processing

To eliminate the influence of the numerical range between different attributes, it is necessary to solve the comparability between indicators through data standardization. Data standardization is to scale down the data to a smaller specific interval. In this paper, the data is uniformly mapped to [-1,1]. Standardization can not only improve the convergence speed and accuracy of the model but also prevent gradient explosions.

$$x' = 2 * \frac{x - x_{min}}{x_{max} - x_{min}} - 1. \quad \textbf{(2)}$$

Equation 2 represents the normalization method. Where x represents the value to be converted, x' is the converted result, x_{max} *and* x_{min} represent the maximum and minimum boundaries of the attribute values, respectively.

2.2 Neural Network

A neural network is a feed-forward network composed of an input layer, an output layer, and a number of hidden layers. In the neural network shown in Figure 1, the input layer is composed of d neurons, each representing an independent variable. There can be several hidden layers in the middle, and the amount of neurons in each hidden layer is set as required, assuming the number is s. The output layer consists of n neurons, and n is the final number of dependent variables to be output. The neurons are connected as follows: there is no connection between neurons in the same layer, and there is a connection between each neuron in two adjacent layers.

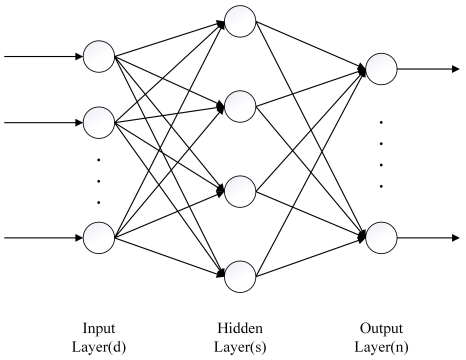


Figure 1. Neural network structure diagram

Figure 2 shows how a single neuron works. Where x is the output value of the previous layer of neurons. w is the weight between connected neurons, and its absolute magnitude represents the effect of the input signal on the neuron, with a positive or negative sign representing enhancement or inhibition of the input signal. f denotes the activation function, which can nonlinearize the neural network. θ is the bias, which allows the activation function to be shifted left or right on the axis, and the use of the bias can enhance the fit of neural networks.

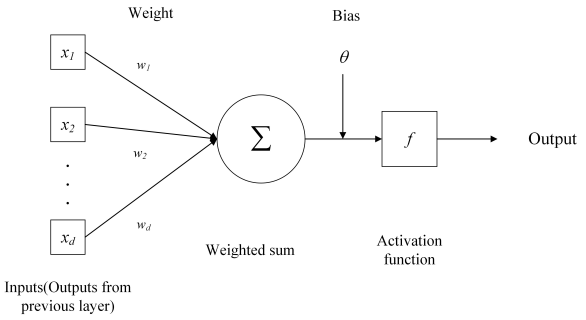


Figure 2. Working diagram of a neuron

The neuron works in the following way: it accepts the weighted output values of each neuron from the previous layer as input values for this neuron. It then uses activation functions and bias for nonlinearization. Finally, it passes these values to the next layer of neurons.

$$a_j = f\left(\sum_{i=1}^{d} w_{ij} x_i - \theta_j\right) \quad (i = 1,2,....,d; j = 1,2,....,s). \quad \textbf{(3)}$$

As an example, take the neural network shown in Figure 1. Equation 3 shows the process of passing data from the input layer to the hidden layer. Where a_j indicates the output value of the jth neuron of the hidden layer, d and s represent the number of neurons in the input layer and neurons in the hidden layer, respectively.

$$y_k = f\left(\sum_{j=1}^{s} w_{jk} a_j - \theta_k\right) \quad (j = 1,2,....,s; k = 1,2,....,n). \quad \textbf{(4)}$$

Similarly, Equation 4 shows the process of passing data from the hidden layer to the output layer. Where y_k denotes the output value of the kth neuron of the output layer, s and n represent the number of neurons in the hidden layer and the output layer, respectively.

However, the ultimate goal is to obtain highly accurate output values. Therefore, a back propagation algorithm is used to continuously adjust the neural network in order to achieve a reduction in the loss function. In this paper, the mean square errors is used as the loss function to evaluate the performance of the neural network.

$$E = \frac{1}{2}\sum_{k=1}^{n}(y_k - y_k^*)^2. \quad \textbf{(5)}$$

The formula for the loss function is given in Equation 5. Where y is the output value and y^* is the actual value.

$$\begin{cases} \Delta w_{jk} = -\eta \dfrac{\partial E}{\partial w_{jk}} = \eta \delta_{jk} a_j (j = 1,2,...,s; k = 1,2,...,n) \\ \Delta w_{ij} = -\eta \dfrac{\partial E}{\partial w_{ij}} = -\eta \dfrac{\partial E}{\partial a_j} \dfrac{\partial a_j}{\partial w_{ij}} = \eta \delta_{ij} x_i (i = 1,2,...,d; j = 1,2,...,s) \end{cases} . \quad \textbf{(6)}$$

The weights can then be adjusted according to the loss function using a backpropagation algorithm, calculated as shown in Equation 6. Where Δw is the value of the change in

weights, δ is the error backpropagation signal, and η is the learning rate. The above is the working process of the neural network, but the training speed of the basic BP algorithm is too slow for practical applications. Therefore, it is necessary to use evolutionary algorithms to optimize the initial thresholds and weights of neural networks in order to improve their efficiency.

3 Salp Swarm Algorithm

Salp swarm algorithm (SSA) is a new swarm intelligence optimization algorithm for solving various optimization problems. SSA emulates the foraging behavior of salps in biology. Salp belongs to the Salpidae family, which is a community-dwelling organism in the deep sea. Unlike other social creatures, salp often forms a swarm called the salp chain, which can help salp forage better. In SSA, the chain is split into two parts: leaders and followers. The individual at the top of the chain is the leader and the others are considered followers. While foraging, the leader searches the space for food and followers follow the leader directly or indirectly.

$$X^i = (X_1^i, X_2^i, ..., X_{dim}^i) \quad (i = 1, 2, ..., n). \tag{7}$$

Mathematically, the position of each salp is defined as shown in Equation 7. Where *dim* is the dimension of the search space, n is the amount of salps. In addition, salp needs to constantly change positions when foraging.

$$X_j^1 = \begin{cases} F_j + c_1((ub_j - lb_j)c_2 + lb_j) & c_3 \geq 0 \\ F_j - c_1((ub_j - lb_j)c_2 + lb_j) & c_3 < 0 \end{cases}. \tag{8}$$

Equation 8 is the leader position update equation, where X_j^1 and F_j indicate the position of the leader and the food source in each dimension, respectively, and F_j also represents the optimal solution of the objective function in the *jth* dimension. But during the foraging process, the location of the food is uncertain. In other words, the global optimal value of the problem to be solved is not known. In this case, suppose that the optimal solution (optimal fitness value) achieved so far is the food source. ub_j and lb_j denote the upper and lower bounds of the search space, respectively. The parameters c_2 and c_3 are random numbers in the [0,1] interval. In fact, they determine where the leader will move next and how far to go. c_1 varies with the number of iterations.

$$c_1 = 2e^{-(\frac{4l}{L})^2}. \tag{9}$$

In Equation 9, l is the current iteration and L is the maximum number of iterations. Equation 8 and Equation 9 show how leaders update their positions based on the optimal solution. c_1 is the most significant factor in SSA, and its value decreased adaptively in the iteration process. Therefore, c_1 plays the role of balancing exploration and production.

After the position of leader is updated, the followers should move together.

$$X_j^i = \frac{1}{2}(X_j^i + X_j^{i-1}). \tag{10}$$

Equation 10 shows how the followers' positions are updated. Where $i \geq 2$, X_j^i shows the position of *ith* follower salp in *jth* dimension. In the formula it can be derived that the follower is directly or indirectly following the leader.

Therefore, the foraging process can be turned into a process of searching for the optimal solution. The simplified steps of SSA are as follows. First, initialize the feasible solution of each individual. Secondly, calculate the fitness value for each individual. The solution of the individual with the most suitable fitness value is taken as the *F*. Finally, the individual solution are continuously updated to find the optimal solution until the end condition is satisfied. The pseudocode for SSA is shown in Algorithm 1, where *N* is the number of individuals.

Algorithm 1. Pseudo code for Salp Swarm Algorithm (SSA)

```
Initialize the feasible solution of N individuals, the current iteration
  number 1, and the maximum iteration number L.
Calculate the fitness value for each individual.
The solution of the individual with the most suitable fitness value was
  used as the F.
while 1 ≤ L do
    for i = 1 to N do
        if i==1 then
            Update the solution of the leader according to the Equation (8).
        end if
        Update the solution of the follower according to the Equation (10).
        Calculate the fitness value for each individual.
        Update the F.
    end for
    1 = 1 + 1
end while
```

4 Parallel Salp Swarm Algorithm

In a computer system, parallelism means that multiple tasks are performed together at the same time. Therefore the same result can be gotten in a shorter period. This paper proposes a parallel salp swarm algorithm (PSSA), in which multiple populations can not only find the optimal value at the same time but also exchange information according to the communication strategy. This method not only increases the diversity of populations but also promotes cooperation between groups. Parallel algorithms that communicate with each other usually converge faster than the original algorithm and achieve smaller optimums. To achieve parallelism, the entire population is first divided into several sub-populations. Each sub-population then searches according to the SSA. When a set communication stimulus point is reached, the obtained information will be exchanged between the sub-populations. Finally, the optimal solution from all sub-populations will be used as a solution. Algorithm 2 is a new communication strategy proposed in this paper.

Algorithm 2. A pseudo-code of communication strategy

```
if meet the communication conditions    then
    Generate random number C.
    if C ≥ 0.5 then / / strategy 1
        for i = np/2 to np do / / np is the number of member in each group.
            Update X_i according to Equation (11).
        end for
    else    / / strategy 2
        Randomly select a follower to exchange identities with the leader.
    end if
end if
```

When the communication conditions are met, a random number C is generated. If $C \geq 0.5$, then strategy 1 is executed, otherwise strategy 2 is executed. Strategy 1 is to mutate the values of some followers based on the obtained optimal value.

$$X_i(t+1) = W * (F + F_g + X_{i-1}(t) + X_i(t)) . \quad \textbf{(11)}$$

Algorithm 3. A Pseudo code for Parallel Salp Swarm Algorithm (PSSA)

```
Initiate the feasible solution of N individuals, the current iteration number 1, and the maximum iteration number L.
Set exchanging time m. / / communication once every m iterations.
Evenly distribute N individuals into G group.
Calculate the fitness value of each individual.
Define the most suitable fitness value in the group g as Fg(g = 1, 2, …, G) and the most suitable fitness value in Fg as F.
while l ≤ L do
    for g = 1 to G do
        for i = 1 to N/G do
            if i == 1 then
                Update the solution of the leader according to the Equation 8.
            end if
            Update the solution of the follower according to the Equation 10.
            Calculate the fitness value of each individual.
            Update F and Fg.
        end for
    end for
    if mod(l,m) =0 then
        communication according to Algorithm 2.
        Update F and Fg.
    end if
    l = l + 1
end while
```

The update method is shown in Equation 11. Where $X_i(t+1)$ denotes the position of the *ith* individual at $t+1$ iteration. F represents the position of the global food source, F_g represents the position of the food source in the same group. In fact, F is the global optimal solution and F_g is the group optimal solution. W is a number that linearly decreases from 0.25 to 0.05 according to iteration. This approach uses F as well as F_g to perturb the position of the follower, changing the situation where the follower can only passively follow the leader and increasing its ability to develop local optimal solutions.

Strategy 2 is to randomly select a follower in the group to exchange identities with the leader. This approach allows each follower the opportunity for global exploration. At the same time, the leader becomes a follower and can lead the surrounding followers to better explore the space around F.

The steps of PSSA are as follows. First, initialize the feasible solution of N individuals in the domain and distribute them evenly among G groups. The solution of the individual in each group with the most suitable fitness value was labeled as F_g *(g = 1, 2, ..., G)*. Label the most suitable of F_g as F. Algorithm 1 is then used to cycle through the search for the optimal solution. When the communication conditions are met, the communication between groups is carried out according to the algorithm 2. Finally, F is the optimal solution of the problem. The pseudo code of PSSA is shown in the algorithm 3.

5 Performance Analysis of Algorithms

In order to test the performance of the proposed PSSA, this paper uses the benchmark function in [25] for illustration. This section will explain in three aspects: test function, parameter setting and experimental results.

5.1 Test Function

Table 4 to Table 6 in the appendix describes the details of the test function, where domain is the range of independent variables, the dimensions is the number of independent variables, and Opt represents the theoretical optimal value of the test function. The test function set consists of three parts: F1-F7 are unimodal functions, F8-F13 are multimodal functions, and F14-F23 are composite benchmark functions. These functions can fully test the performance of the algorithm. In addition, PSSA is compared to SSA, PSO, and PPSO. Where PPSO is the parallel particle swarm optimization.

5.2 Parameter Setting

The parameter settings of the four algorithms are shown in Table 1. The test parameters are set as follows: the number of population is 80, and the maximum number of iterations is 200. For the algorithm to join the parallel strategy, this paper set the exchange every 10 iterations. Each test function is run 30 times, and then the average and variance are calculated as evaluation criteria.

Table 1. Parameter setting

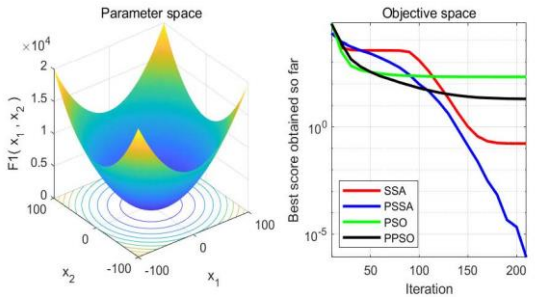
Algorithm	Parameters setting
SSA/PSSA	c_2 and c_3 are random values in [1, 0]
PSO/PPSO	$V_{max} = 6$, $W_{max} = 0.9$, $c_1 = 2$, $c_2 = 2$

5.3 Results and Discussion

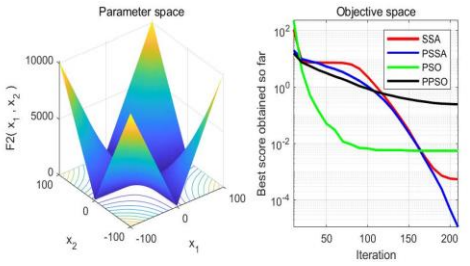
Table 2 shows the best fitness values obtained by the four algorithms of SSA, PSSA, PSO and PPSO after 200 iterations. The ones marked in bold in the table are the optimal values of the winning algorithm. It can be seen that the PSSA has reached the minimum 13 times in total. Therefore, PSSA is excellent in finding the best. Figure 3 to Figure 5 shows the convergence curves of the four algorithms in each benchmark function. F1-F7 in Table 4 are unimodal functions, and there is only one global optimal solution. As can be seen in Figure 3, for the unimodal function, although the convergence speed of PSSA becomes slower, the minimum adaptive value can be reached in the same number of iterations. F8-F13 in Table 5 are multimodal functions, which has several local optimal solutions and a global optimal solution. It is usually used to test whether the algorithm can avoid falling into the local optimal solution. As shown in Figure 4, PSSA can bypass the local optima and obtain the optimal solution with higher accuracy. F14-F23 in Table 6 are test functions for other peak types. As can be seen from Figure 5, PSSA is slightly better than the other three algorithms. For F16-F22, PSSA can always find the best solution together with other algorithms. In general, the performance of PSSA is good.

Table 2. The results of simulation experiments

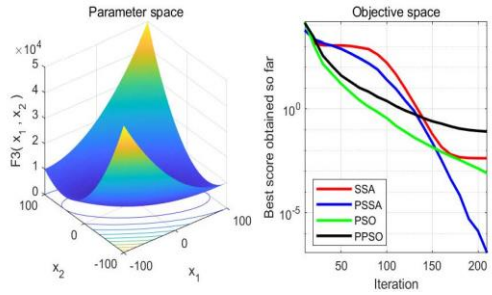
Function	SSA		PSSA		PSO		PPSO	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	1.68E-01	1.79E-01	**8.61E-07**	3.42E-06	2.12E+02	1.17E+02	2.01E+01	7.46E+00
F2	5.43E-04	2.50E-03	**1.09E-05**	5.95E-06	5.60E-03	2.77E-02	2.51E-01	2.39E-01
F3	4.26E-03	1.74E-02	**1.21E-07**	3.45E-07	8.02E-04	8.98E-04	8.19E-02	8.18E-02
F4	**2.23E-05**	8.87E-06	2.82E-05	2.40E-05	9.82E-02	3.55E-01	1.59E-01	1.02E-01
F5	9.23E+01	1.33E+02	**7.80E+00**	1.32E+00	1.05E+01	1.75E+01	1.36E+01	2.63E+01
F6	**1.05E-09**	3.65E-10	5.50E-09	3.78E-09	5.89E-05	3.22E-04	2.98E-03	4.84E-03
F7	1.11E-02	7.50E-03	**2.31E-04**	2.61E-04	2.28E-03	1.76E-03	8.14E-03	4.84E-03
F8	-2.85E+03	3.07E+02	**-3.08E+03**	2.51E+02	-2.56E+03	3.18E+02	-2.49E+03	3.21E+02
F9	1.09E+01	5.39E+00	**4.81E+00**	2.53E+00	8.79E+00	3.63E+00	6.84E+00	3.80E+00
F10	7.97E-01	1.11E+00	**1.04E-05**	3.28E-06	4.18E-01	6.12E-01	6.67E-01	5.64E-01
F11	1.79E-01	8.28E-02	**3.23E-02**	2.50E-02	1.48E-01	6.77E-02	3.80E-01	1.56E-01
F12	6.26E-01	9.22E-01	**1.85E-02**	5.96E-02	1.14E-01	2.65E-01	5.46E-02	1.22E-01
F13	3.63E-03	5.91E-03	**7.28E-04**	9.91E-04	4.03E-03	1.81E-02	8.27E-03	8.89E-03
F14	1.46E+00	9.28E-01	**1.23E+00**	5.01E-01	1.30E+00	4.63E-01	1.99E+00	1.22E+00
F15	1.45E-03	3.59E-03	**6.27E-04**	2.68E-04	1.15E-03	3.64E-03	7.67E-04	5.67E-04
F16	-1.03E+00	4.15E-14	-1.03E+00	4.30E-14	-1.03E+00	6.65E-16	-1.03E+00	6.12E-16
F17	3.98E-01	7.12E-14	3.98E-01	8.78E-14	3.98E-01	0.00E+00	3.98E-01	0.00E+00
F18	3.00E+00	3.54E-13	3.00E+00	3.92E-13	3.00E+00	1.40E-15	3.00E+00	1.77E-15
F19	-3.86E+00	1.66E-08	-3.86E+00	1.61E-05	-3.82E+00	2.36E-02	-3.86E+00	2.52E-15
F20	-3.22E+00	7.01E-02	-3.31E+00	4.18E-02	-2.28E+00	3.81E-01	**-3.22E+00**	2.17E-02
F21	-7.81E+00	3.22E+00	-1.02E+01	5.67E-11	-2.44E+00	1.25E+00	-1.02E+01	2.21E-11
F22	-8.08E+00	3.39E+00	-1.04E+01	5.13E-11	-2.28E+00	1.26E+00	-1.04E+01	6.39E-15
F23	-9.65E+00	2.31E+00	-1.05E+01	4.74E-11	-2.26E+00	1.01E+00	-1.05E+01	1.82E-13
Win	2		13		0		1	



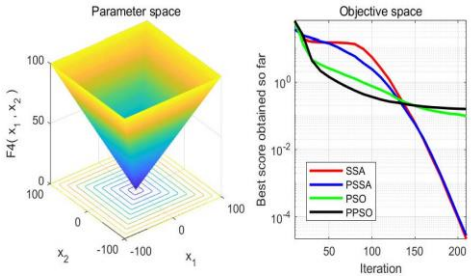
(a) F1



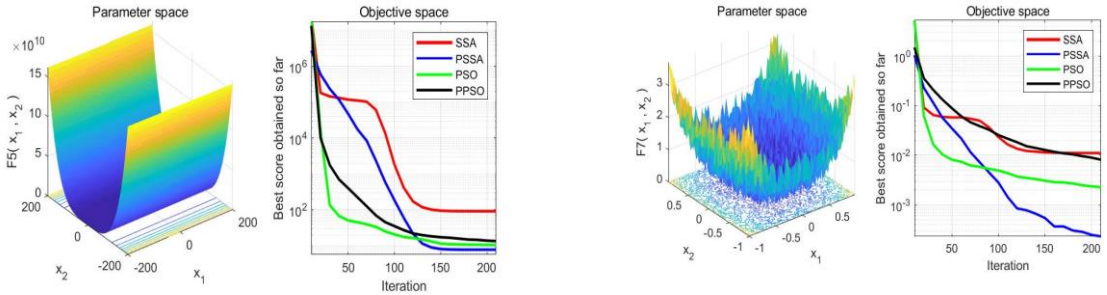
(b) F2



(c) F3

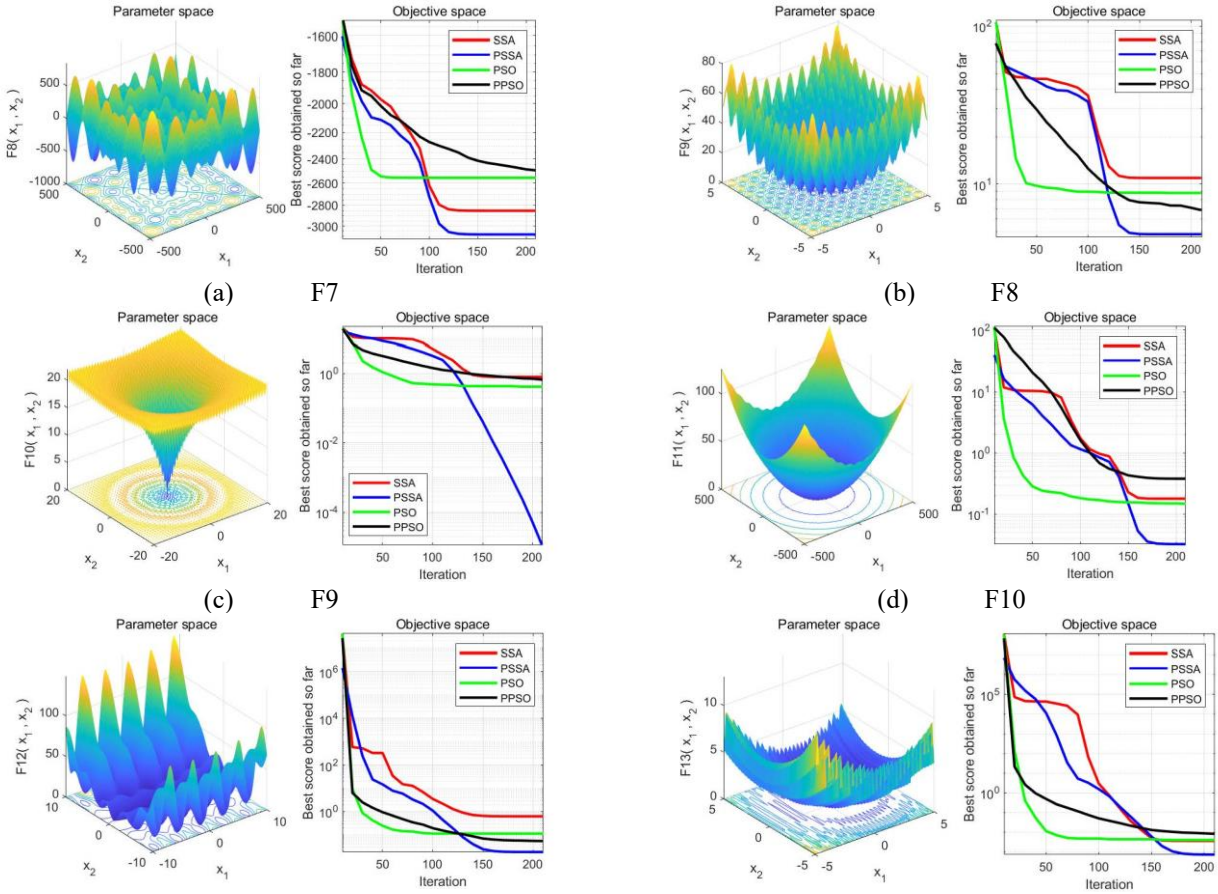


(d) F4



(e) F5 (f) F6

Figure 3. Convergence curves of benchmark functions

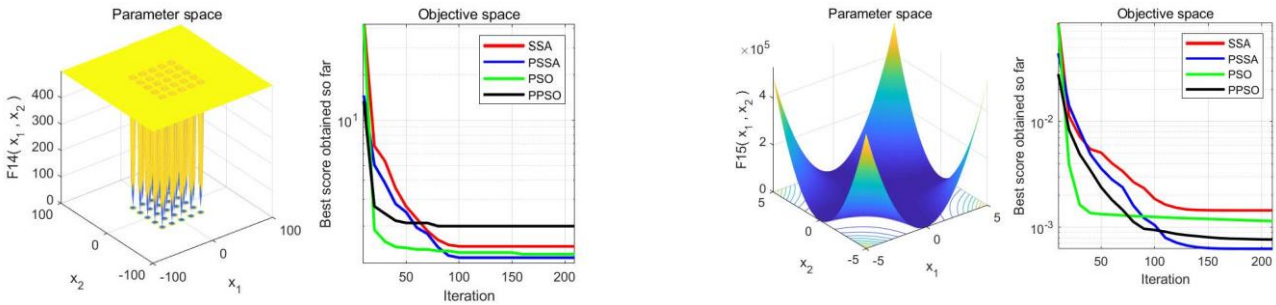


(a) F7 (b) F8

(c) F9 (d) F10

(e) F11 (f) F12

Figure 4. Convergence curves of benchmark functions



(a) F14 (b) F15

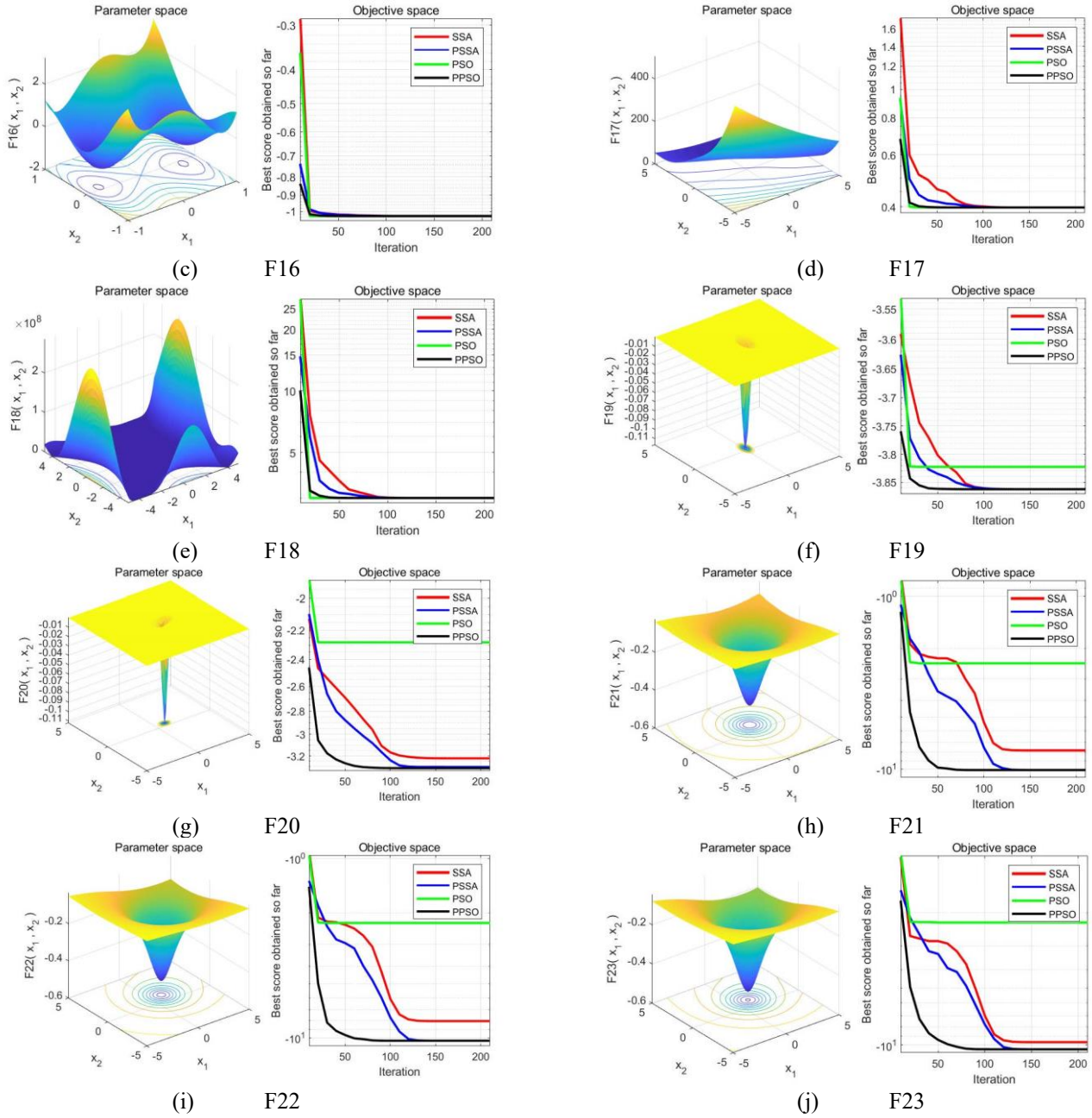


Figure 5. Convergence curves of benchmark functions

6 Applied PSSA Based Neural Network for Power Load Forecast

The neural network fits the nonlinear relationship between the input vector and the output vector by learning training data, so as to realize the prediction of future data. Therefore, the performance of the neural network determines the accuracy of load forecasting. This section describes how to use PSSA to optimize a neural network and use the optimized neural network for load forecasting. In addition, this section also adds Sine Cosine Algorithm (SCA), Whale Optimization Algorithm (WOA), and their parallel versions PSCA and PWOA to compare the proposed PSSA. The experimental results show that the hybrid model of PSSA and neural network performs better in load forecasting.

6.1 Proposed BPNN Combined with the PSSA

The process of PSSA optimizing neural network is divided into two stages. The first stage is to optimize the structure of the neural network. This paper is based on a single hidden layer neural network. In the optimization algorithm, each individual represents the number of hidden layer neurons and then iterates until it converges to the optimal number of neurons. A sigmoid transfer function was used for the hidden layers, the learning rate of the neural network is set to 0.1, and the target error is 0.001. In addition, the initial values of the weights and thresholds have an important impact on the performance of the neural network. Therefore, the second stage is to optimize the initial weights and thresholds of the neural network. At this stage, each individual represents the weight and threshold components. Later, in the iterative

process, by constantly adjusting the size of the weights and thresholds, the neural network prediction performance is better. Finally, the best initial weight and threshold are obtained. Figure 6 shows the flow chart of PSSA and neural network cooperation.

As can be seen in the flowchart, this paper uses the average error between the output value and the true value as a fitness function.

$$f = \left(\sum_{i=1}^{n} \left| y_i - y_i^* \right| / y_i \right) / n \ . \tag{12}$$

Equation 12 is the calculation function for the fitness value. Where y represents the actual result, y^* is the output of the neural network, and n is the amount of training data.

6.2 Data Information

In order to ensure the accuracy of the hybrid model, this paper uses the standard data set provided by the 9th Modeling Competition [57], which selects the data set "China Electrical Engineering Society Cup" National Undergraduate Electrical Engineering Mathematical from January 1, 2012 to June 30, 2014 Load data of a certain area in China. In addition, the data set contains temperature, relative humidity, rainfall, the number of days of the week, whether it is a holiday, and the load value of a day. The load value of a day is the output vector, which has one component in total. The rest of the information is the input vector, with a total of 8 components. This article uses the data from January 1, 2012 to May 31, 2014, as the training set, and uses the data from June 1, 2014 to June 30, 2014, as the test set to test the performance of the hybrid model. As shown in the next section, the data obtained show that the proposed model can accurately predict the load.

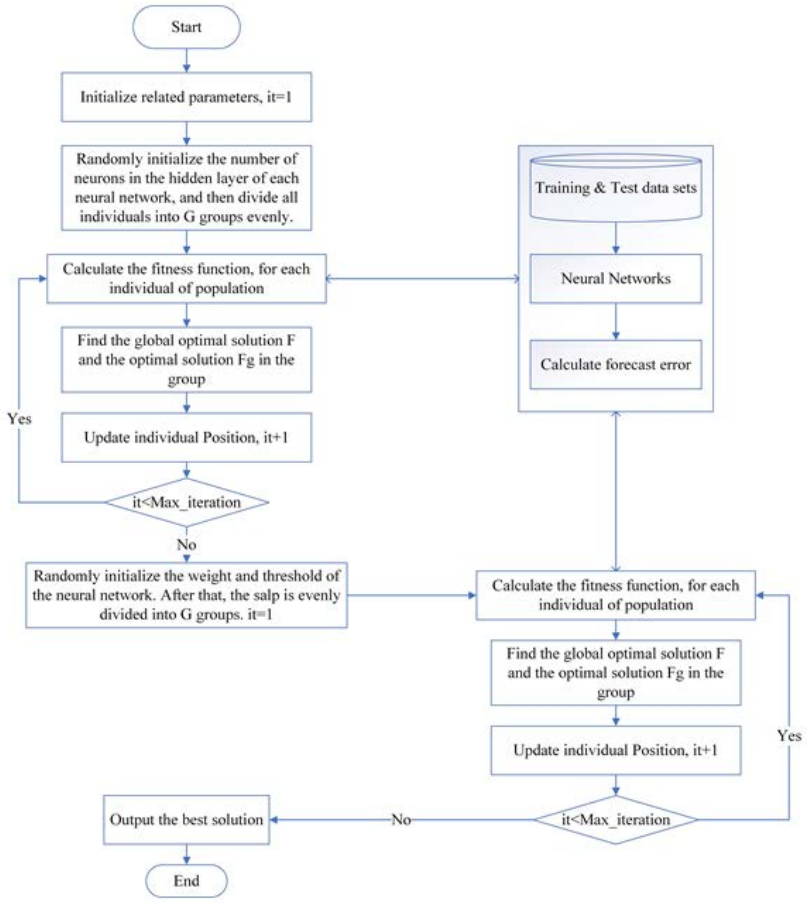


Figure 6. PSSA optimized neural network flow chart

6.3 Results and Discussion

First, PSSA is used to optimize the structure of the neural network. All structures have 8 inputs and 1 output, but the number of neurons in the hidden layer is different.

The optimal number of neurons in the hidden layer is 48 in the final training. After determining the structure of the neural network, use PSSA to optimize the relevant weights and thresholds, and get a neural network with the best performance. Finally, the neural network is used to predict the load from June 1, 2014, to June 30, 2014. Figure 7 shows the actual load and the result of using neural network and hybrid model prediction, where the x-axis represents time and the y-axis represents the load value. It can be seen that the optimized neural network is more accurate than NN. However, the neural

network optimized by SSA has a large deviation in predicting the load on the 6th day and the 18th day. In contrast, the prediction curve of PSSA and neural networks is relatively flat and is more in line with the real load value. It can be seen that, compared with SSA, the optimization result of PSSA is better. Figure 8 shows the prediction results of four parallel algorithms combined with neural networks. It can be seen that among the load curves predicted by the four models, the curve of PSSA is sandwiched in the middle and is closer to the actual load value.

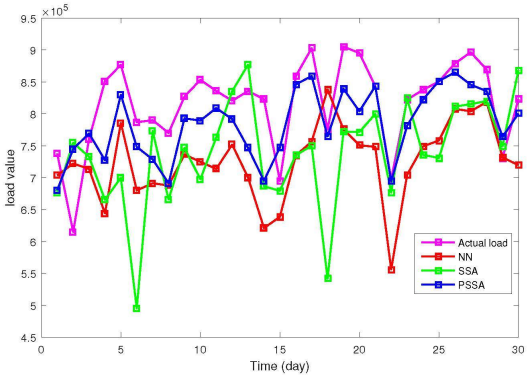


Figure 7. The forecast results of NN, SSA, PSSA

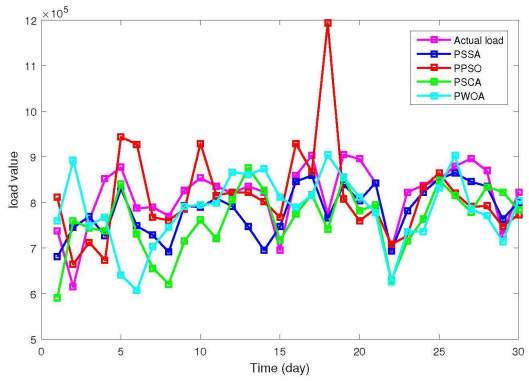


Figure 8. The forecast results

Table 3 shows the average prediction accuracy of all models. It can also be seen that the optimized neural network has better performance. Moreover, compared with other optimization algorithms, PSSA can achieve better prediction accuracy.

Table 3. The prediction results

Method	Prediction accuracy (%)
NN	87.64
SSA	88.44
PSSA	94.14
PSO	88.76
PPSO	91.48
SCA	89.56
PSCA	90.79
WOA	88.24
PWOA	90.22

Figure 9 shows the prediction errors of the four mixed models with the highest prediction accuracy. It can be seen that the neural network optimized by PSSA has a relatively flat error curve. And the prediction error is closer to the zero horizontal line. Therefore, it can be concluded that PSSA can better optimize the neural network to obtain better prediction accuracy.

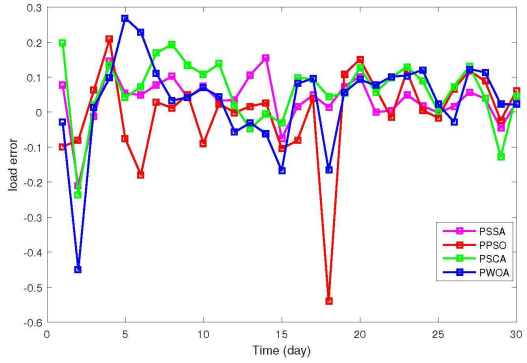


Figure 9. The prediction error of four algorithms

7 Conclusions and Future Work

Accurate load forecasting enables power companies to reasonably schedule the start and stop of generators, maintain the safety and stability of grid operation, and have a good impact on the economy and society. However, neural networks have the disadvantages of slow speed and insufficient accuracy in predicting load. Therefore, this paper proposes a parallel salp swarm algorithm and uses it to optimize the structure and parameters of the neural network. First, 23 test functions are used to test the performance of PSSA. The experimental results show that PSSA performs well in exploration and development. And the convergence speed of PSSA is faster than SSA, but it still needs to be improved to further improve the performance of the algorithm. After that, four optimization algorithms are used to optimize the structure and parameters of the neural network. Finally, the optimized neural network is used to make predictions, and the results show that the hybrid model of PSSA and neural networks can achieve the highest accuracy.

Nevertheless, the following work is still needed in the future. First, continue to improve the accuracy of forecasts. Second, study the influence of other hyperparameters in the neural network on prediction. Third, discuss the impact of different data on the performance of neural networks. Fourth, since the parallelism in this article is only theoretical parallelism, it can be implemented on multi-processors next.

8 Appendix

This appendix introduces the test functions in the literature [25]. The details of these test functions are shown in Table 4 to Table 6. It consists of three parts: F1-F7 in Table 4 are unimodal functions, these functions have only one optimum and there are no local optima. These types of search spaces are appropriate for testing the convergence speed. F8-F13 in Table 5 are multimodal functions, and F14-F23 in Table 6 are composite benchmark functions. These functions have more than one optimum, which make them suitable for benchmarking the local optima avoidance and explorative behaviour of optimization algorithms.

Table 4. Single-peak test function

Number	Function	Domain	Dimension	Opt
1	$F_1(z)=\sum_{j=1}^{M} z_j^2$	[-100, 100]	30	0
2	$F_2(z)=\sum_{j=1}^{M}\lvert z_j\rvert+\prod_{j=1}^{M}\lvert z_j\rvert$	[-10, 10]	30	0
3	$F_3(z)=\sum_{j=1}^{M}\left(\sum_{k=1}^{j} z_k\right)^2$	[-100, 100]	30	0
4	$F_4(z)=max_j\lvert z_j\rvert, j\in[1,m]$	[-100, 100]	30	0
5	$F_5(z)=\sum_{j=1}^{M-1}\left[100\left(z_{j+1}-z_j^2\right)^2+\left(z_j-1\right)^2\right]$	[-30, 30]	30	0
6	$F_6(z)=\sum_{j=1}^{M}\left(\left[z_j+0.5\right]\right)^2$	[-100, 100]	30	0
7	$F_7(z)=\sum_{j=1}^{M} j*z_j^2+rand[0,1)$	[-1.28, 1.28]	30	0

Table 5. Multi-peak test function

Number	Function	Domain	Dimension	Opt
8	$F_8(z)=\sum_{j=1}^{M}-z_j*sin\left(\sqrt{\lvert z_j\rvert}\right)$	[-500, 500]	30	-12569
9	$F_9(z)=\sum_{j=1}^{M}\left[z_j^2-10*cos\left(2\pi z_j\right)+10\right]$	[-5.12, 5.12]	30	0
10	$F_{10}(z)=-20*exp\left(-0.2\sqrt{\frac{1}{M}\sum_{j=1}^{M} z_j^2}\right)$	[-32, 32]	30	0
11	$F_{11}(z)=\frac{1}{4000}*\sum_{j=1}^{M} z_j^2-\prod_{j=1}^{M} cos\left(\frac{z_j}{\sqrt{j}}\right)+1$	[-600, 600]	30	0
12	$F_{12}(z)=\frac{\pi}{M}*\{10*sin\left(\pi z_1\right)+\sum_{j=1}^{M-1}\left(z_j-1\right)^2\left[1+10*sin^2(\pi z_{j+1})\right]+(z_M-1)^2\}+\sum_{j=1}^{M} u(z_j,10,100,4),$ $z_j=1+\frac{z_j+1}{4}*u(z_l,a,k,m)=\begin{cases}k(z_j-a), z>a\\0, -a<z_j<a\\k(-z_j-a), z>a\end{cases}$	[-50, 50]	30	0
13	$F_{13}(z)=0.1*\{sin^2\left(3\pi z_1\right)+\sum_{j=1}^{M}\left(z_j-1\right)^2\left[1+sin^2\left(3\pi z_j+1\right)\right]+\left(z_M-1\right)^2\left[1+sin^2\left(2\pi z_M\right)\right]\}$	[-50, 50]	30	0

Table 6. Others-peak test function

Number	Function	Domain	Dimension	Opt
14	$F_{14}(z)=\left(\frac{1}{500}*\sum_{j=1}^{25}\frac{1}{j+\sum_{k=1}^{2}\left(z_k-a_{kj}\right)^6}\right)^{-1}$	[-65, 65]	2	1
15	$F_{15}(z)=\sum_{j=1}^{11}\left[a_j-\frac{z_1\left(b_j^2+b_jz^2\right)}{b_j^2+b_jz^3+z^4}\right]^2$	[-5, 5]	4	0.0003
16	$F_{16}(z)=4z_j^2-2.1z_j^4+\frac{1}{3}z_j^6+z_jz_2-4z_2^2+4z_2^4$	[-5, 5]	2	-1.0316
17	$F_{17}(z)=\left(z_2-\frac{5.1}{4\pi^2}z_j^2+\frac{5}{\pi}z_j-6\right)^2$	[-5, 5]	2	0.398
18	$F_{18}(z)=[1+\left(z_1+z_2+j\right)^2*$ $\left(19-14z_1+3z_1^2-14z_2+6z_1z_2+3z_2^2\right)]$ $*[30+\left(2z_1-3z_2\right)^2*$ $\left(18-32z_1+12z_1^2+48z_2-36z_1z_2+27z_2^2\right)]$	[-2, 2]	2	3
19	$F_{19}(z)=-\sum_{j=1}^{4}c_j*exp\left(-\sum_{k=1}^{3}a_{jk}\left(z_k-p_{jk}\right)^2\right)$	[1, 3]	30	-3.86
20	$F_{20}(z)=-\sum_{j=1}^{4}c_j*exp\left(-\sum_{k=1}^{6}a_{jk}\left(z_k-p_{jk}\right)^2\right)$	[0, 1]	6	-3.32
21	$F_{21}(z)=-\sum_{j=1}^{5}\left[\left(z-a_j\right)\left(z-a_j\right)^T+c_j\right]^{-1}$	[0, 10]	4	-10.1532
22	$F_{22}(z)=-\sum_{j=1}^{7}\left[\left(z-a_j\right)\left(z-a_j\right)^T+c_j\right]^{-1}$	[0, 10]	4	-10.4028
23	$F_{23}(z)=-\sum_{j=1}^{10}\left[\left(z-a_j\right)\left(z-a_j\right)^T+c_j\right]^{-1}$	[0, 10]	4	-10.5363

References

[1] D. Wang, H. Luo, O. Grunder, Y. Lin, H. Guo, Multi-step ahead electricity price forecasting using a hybrid model based on two-layer decomposition technique and BP neural network optimized by firefly algorithm, *Applied Energy*, Vol. 190, pp. 390-407, March, 2017.

[2] C. Sun, D. Gong, Support vector machines with PSO algorithm for short-term load forecasting, *2006 IEEE International Conference on Networking, Sensing and Control*, Ft. Lauderdale, FL, USA, 2006, pp. 676-680.

[3] D. Niu, Y. Wang, D. Wu, Power load forecasting using support vector machine and ant colony optimization, *Expert Systems with Applications*, Vol. 37, No. 3, pp. 2531-2539, March, 2010.

[4] Y. Wang, D. Niu, X. Ma, Optimizing of SVM with hybrid PSO and genetic algorithm in power load forecasting, *Journal of Networks*, Vol. 5, No. 10, pp. 1192-1200, October, 2010.

[5] N. Abu-Shikhah, F. Elkarmi, Medium-term electric load forecasting using singular value decomposition, *Energy*, Vol. 36, No. 7, pp. 4259-4271, July, 2011.

[6] T. Yalcinoz, U. Eminoglu, Short term and medium term power distribution load forecasting by neural networks, *Energy Conversion and Management*, Vol. 46, No. 9-10, pp. 1393-1405, June, 2005.

[7] M. Gavrilas, I. Ciutea, C. Tanasa, Medium-term load forecasting with artificial neural network models, *16th International Conference and Exhibition on Electricity Distribution*, Amsterdam, Netherlands, 2001, pp. 1-5.

[8] J. Zhang, Y.-M. Wei, D. Li, Z. Tan, J. Zhou, Short term electricity load forecasting using a hybrid model, *Energy*, Vol. 158, pp. 774-781, September, 2018.

[9] Z. Xiao, S.-J. Ye, B. Zhong, C.-X. Sun, BP neural network with rough set for short term load forecasting, *Expert Systems with Applications*, Vol. 36, No. 1, pp. 273-279, January, 2009.

[10] W. Charytoniuk, M.-S. Chen, Very short-term load forecasting using artificial neural networks, *IEEE transactions on Power Systems*, Vol. 15, No. 1, pp. 263-268, February, 2000.

[11] M. Talaat, M. Farahat, N. Mansour, A. Hatata, Load forecasting based on grasshopper optimization and a multilayer feed-forward neural network using regressive approach, *Energy*, Vol. 196, Article No. 117087, April, 2020.

[12] A. Bakirtzis, V. Petridis, S. Kiartzis, M. Alexiadis, A. Maissis, A neural network short term load forecasting model for the Greek power system, *IEEE Transactions on power systems*, Vol. 11, No. 2, pp. 858-863, May, 1996.

[13] M. Barman, N. D. Choudhury, S. Sutradhar, A regional hybrid GOA-SVM model based on similar day approach for short-term load forecasting in Assam, India, *Energy*, Vol. 145, pp. 710-720, February, 2018.

[14] A. Selakov, D. Cvijetinovic, L. Milovic, S. Mellon, D. Bekut, Hybrid PSO--SVM method for short-term load forecasting during periods with significant temperature variations in city of Burbank, *Applied Soft Computing*, Vol. 16, pp. 80-88, March, 2014.

[15] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, Y. Zhang, Short-term residential load forecasting based on LSTM recurrent neural network, *IEEE Transactions on Smart Grid*, Vol. 10, No. 1, pp. 841-851, January, 2019.

[16] S. Muzaffar, A. Afshari, Short-term load forecasts using LSTM networks, *Energy Procedia*, Vol. 158, pp. 2922-2927, February, 2019.

[17] Q.-W. Chai, S.-C. Chu, J.-S. Pan, W.-M Zheng, Applying Adaptive and Self Assessment Fish Migration Optimization on Localization of Wireless Sensor Network on 3-D Terrain, *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 11, No. 2, pp. 90-102, June, 2020.

[18] S. C. Chu, X. Xue, J. S. Pan, X. Wu, Optimizing ontology alignment in vector space, *Journal of Internet Technology*, Vol. 21, No. 1, pp. 15-22, January, 2020.

[19] J.-S. Pan, X. Wang, S.-C. Chu, T.-T. Nguyen, A multi-group grasshopper optimisation algorithm for application in capacitated vehicle routing problem, *Data Science and Pattern Recognition*, Vol. 4, No. 1, pp. 41-56, July, 2020.

[20] J. Kennedy, R. Eberhart, Particle swarm optimization, *Proceedings of ICNN'95-International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp. 1942-1948.

[21] R. C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, Seoul, Korea (South), 2001, pp. 81-86.

[22] Z.-H. Zhan, J. Zhang, Y. Li, H. S.-H. Chung, Adaptive particle swarm optimization, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 39, No. 6, pp. 1362-1381, December, 2009.

[23] W. Song, C. Cattani, C.-H. Chi, Multifractional Brownian motion and quantum-behaved particle swarm optimization for short term power load forecasting: An integrated approach, *Energy*, Vol. 194, Article No. 116847, March, 2020.

[24] Y. Shi, R. C. Eberhart, Parameter selection in particle swarm optimization, *International conference on evolutionary programming*, San Diego, California, USA, 1998, pp. 591-600.

[25] J. Wang, C. Ju, Y. Gao, A. K. Sangaiah, G.-J. Kim, A PSO based Energy Efficient Coverage Control Algorithm for Wireless Sensor Networks, *Computers Materials & Continua*, Vol. 56, No. 3, pp. 433-446, 2018.

[26] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Advances in engineering software*, Vol. 69, pp. 46-61, March, 2014.

[27] P. Hu, J. S. Pan, S. C. Chu, Improved binary grey wolf optimizer and its application for feature selection, *Knowledge-Based Systems*, Article No. 105746, May, 2020.

[28] T. Jayabarathi, T. Raghunathan, B. Adarsh, P. N. Suganthan, Economic dispatch using hybrid grey wolf optimizer, *Energy*, Vol. 111, pp. 630-641, September, 2016.

[29] M. Dorigo, G. Di Caro, Ant colony optimization: a new meta-heuristic, *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, Washington, DC, USA, 1999, pp. 1470-1477.

[30] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, *IEEE computational intelligence magazine*, Vol. 1, No. 4, pp. 28-39, November, 2006.

[31] J. E. Bell, P. R. McMullen, Ant colony optimization techniques for the vehicle routing problem, *Advanced engineering informatics*, Vol. 18, No. 1, pp. 41-48, January, 2004.

[32] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: J. R. Gonzalez, D. A. Pelta, C. Cruz, G. Terrazas, N. Krasnogor (Eds.), *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Springer, Berlin, Heidelberg, 2010, pp. 65-74.

[33] X.-S. Yang, A. H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, *Engineering computations*, Vol. 29, No. 5, pp. 464-483, July, 2012.

[34] X.-S. Yang, Bat algorithm for multi-objective optimisation, *International Journal of Bio-Inspired Computation*, Vol. 3, No. 5, pp. 267-274, September, 2011.

[35] S.-C. Chu, P.-W. Tsai, J.-S. Pan, Cat swarm optimization, *Pacific Rim international conference on artificial intelligence*, Guilin China, 2006, pp. 854-858.

[36] P.-W. Tsai, J.-S. Pan, S.-M. Chen, B.-Y. Liao, S.-P. Hao, Parallel cat swarm optimization, *2008 international conference on machine learning and cybernetics*, Kunming, 2008, pp. 3328-3333.

[37] P.-W. Tsai, J.-S Pan, S.-M. Chen, B.-Y. Liao, Enhanced parallel cat swarm optimization based on the Taguchi method, *Expert Systems with Applications*, Vol. 39, No. 7, pp. 6309-6319, June, 2012.

[38] Z. Meng, J.-S. Pan, H. Xu, QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm: A cooperative swarm based algorithm for global optimization, *Knowledge-Based Systems*, Vol. 109, pp. 104-121, October, 2016.

[39] Z. Meng, J.-S. Pan, A competitive QUasi-Affine TRansformation Evolutionary (C-QUATRE) algorithm for global optimization, *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, Hungary, 2016, pp. 001644-001649.

[40] Z. G. Du, J. S. Pan, S. C. Chu, H. J. Luo, P. Hu, Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in

wireless sensor networks, *IEEE Access*, Vol. 8, pp. 8583-8594, January, 2020.
[41] J. Zhuang, H. Luo, T.-S. Pan, J.-S. Pan, Improved Flower Pollination Algorithm for the Capacitated Vehicle Routing Problem, *Journal of Network Intelligence*, Vol. 5, No. 3, pp. 141-156, August, 2020.
[42] D. Whitley, A genetic algorithm tutorial, *Statistics and computing*, Vol. 4, No. 2, pp. 65-85, June, 1994.
[43] G. R. Harik, F. G. Lobo, D. E. Goldberg, The compact genetic algorithm, *IEEE transactions on evolutionary computation*, Vol. 3, No. 4, pp. 287-297, November, 1999.
[44] P.-C. Song, J.-S. Pan, S.-C. Chu, A parallel compact cuckoo search algorithm for three-dimensional path planning, *Applied Soft Computing*, Vol. 94, Article No. 106443, September, 2020.
[45] Z. Yuan, W. Wang, H. Wang, S. Mizzi, Combination of cuckoo search and wavelet neural network for midterm building energy forecast, *Energy*, Vol. 202, Article No. 117728, July, 2020.
[46] J. S. Pan, P. C. Song, S. C. Chu, Y. J. Peng, Improved compact cuckoo search algorithm applied to location of drone logistics hub, *Mathematics*, Vol. 8, No. 3, Article No. 333, March, 2020.
[47] K. Vijayalakshmi, P. Anandan, Global levy flight of cuckoo search with particle swarm optimization for effective cluster head selection in wireless sensor network, *Intelligent Automation and Soft Computing*, Vol. 26, No. 2, pp. 303-311, 2020.
[48] K. V. Price, Differential evolution, in: I. Zelinka, V. Snasel, A. Abraham (Eds.), *Handbook of Optimization*, Springer, Berlin, Heidelberg, 2013, pp. 187-214.
[49] Z. Meng, J. S. Pan, K. K. Tseng, PaDE: An enhanced Differential Evolution algorithm with novel control parameter adaptation schemes for numerical optimization, *Knowledge-Based Systems*, Vol. 168, pp. 80-99, March, 2019.
[50] J. S. Pan, N. Liu, S. C. Chu, A hybrid differential evolution algorithm and its application in unmanned combat aerial vehicle path planning, *IEEE Access*, Vol. 8, pp. 17691-17712, January, 2020.
[51] A. Q. Tian, S. C. Chu, J. S. Pan, H. Cui, W. M. Zheng, A compact pigeon-inspired optimization for maximum short-term generation mode in cascade hydroelectric power station, *Sustainability*, Vol. 12, No. 3, Article No. 767, February, 2020.
[52] S. Mirjalili, S. M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, *Neural Computing and Applications*, Vol. 27, No. 2, pp. 495-513, February, 2016.
[53] X. Wang, J. S. Pan, S. C. Chu, A parallel multi-verse optimizer for application in multilevel image segmentation, *IEEE Access*, Vol. 8, pp. 32018-32030, February, 2020.
[54] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, *Advances in Engineering Software*, Vol. 114, pp. 163-191, December, 2017.
[55] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, H. Fujita, An efficient binary salp swarm algorithm with crossover scheme for feature selection problems, *Knowledge-Based Systems*, Vol. 154, pp. 43-67, August, 2018.
[56] S. Ekinci, B. Hekimoglu, Parameter optimization of power system stabilizer via salp swarm algorithm, *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, Istanbul, Turkey, 2018, pp. 143-147.
[57] Electrical Engineering Mathematics Committee of Chinese Society for Electrical Engineering, Topics of the Ninth "Cup of Chinese Society for Electrical Engineering" National College Students Electrical Engineering Mathematical Modeling Contest, 2016, http://shumo.neepu.edu.cn.

Biographies

Jin-Liang Zhou received the B.S. degree from Shandong Agricultural University, China, in 2019. He is currently pursuing the master degree with the Shandong University of Science and Technology, Qingdao, China. His research interests include intelligence algorithms and load forecasting.

Shu-Chuan Chu received the Ph.D. degree in 2004 from the School of Computer Science, Engineering and Mathematics, Flinders University of South Australia. She joined Flinders University in December 2009 after 9 years at the Cheng Shiu University, Taiwan. She is the Research Fellow in the College of Science and Engineering of Flinders University, Australia from December 2009. Currently, She is the Research Fellow with PhD advisor in the College of Computer Science and Engineering of Shandong University of Science and Technology from September 2019. Her research interests are mainly in Swarm Intelligence, Intelligent Computing and Data Mining.

Ai-Qing Tian received his B.S. degree from Taishan Institute Of Technology, Shandong University Of Science and Technology in 2019. He is currently pursuing the master degree with the Shandong University of Science and Technology, Qingdao, China. His recent research interests are swarm intelligence and artificial neural networks.

Yan-Jun Peng received his Dr. degree in march, 2004. He joined the Department of Computer Science, Shandong University of science and technology, Qingdao, China, in 1996, where he was promoted to professor in 2010. His main research interests include image processing, deep learning and visualization.

Jeng-Shyang Pan received the B.S. degree in electronic engineering from the National Taiwan University of Science and Technology in 1986, the M.S. degree in communication engineering from National Chiao Tung University, Taiwan, in 1988, and the Ph.D. degree in electrical engineering from the University of Edinburgh, U.K., in 1996. He is currently the Professor of Shandong University of Science and Technology. He is the IET Fellow, U.K., and has been the Vice Chair of the IEEE Tainan Section.