BP Load Forecasting Algorithm Based on Improved Simulated Annealing Algorithm

Weihua Pan[1, *], Wenjie Wang[1]

[1]School of Control and Computer Engineering, North China Electric Power University, Baoding 071003, Hebei province, China

Abstract

Compared with traditional prediction algorithms, the BP neural network model has strong learning ability and self-adaptation ability, and has strong ability to fit nonlinear data, but the iteration speed is slow and the overall performance is poor; the PSO-BP prediction model improves the prediction accuracy, but it is still easy to mature Converge and fall into a local minimum. This paper proposes a variance representation of population diversity (PA) based on Euclidean distance, and uses the probabilistic jump feature of the simulated annealing algorithm to propose a simulated annealing (PASA) algorithm based on population diversity. Applying this algorithm to the PSO-BP model, the model training results show that the PASA algorithm not only guarantees the diversity and uniform distribution of particle populations, but also uses probabilistic jumps to avoid early rapid convergence, and the prediction accuracy is improved from the average relative error of the PSO model. 4.335% increased to 3.775%.

Keywords

BP neural network; Particle swarm algorithm; Population diversity; Simulated annealing.

1. Introduction

Load forecasting has always been an important issue in power information systems. Accurate load forecasting is an important means of energy conservation and economic efficiency improvement. Nowadays, the power consumption structure shows a diversified trend, and the consumption patterns and consumption methods are changing with each passing day. The complexity of power grid changes makes it more difficult to for-mulate power generation plans, and may even cause an imbalance in power supply and demand. And load forecasting from the perspective of algorithms, provides data support for power plants, so that power generation companies can reasonably formulate power production plans, achieve a basic balance of power supply and demand, and maximize production benefits. Power load forecasting is playing an increasingly important role in the intelligent management of power grids [1].

There are many factors that affect load changes. Load forecasting is not only based on historical load data, but also takes into account the influence of multiple variables such as the level of local economic development, seasonal changes, weather conditions, and holidays, so as to find the mathematical relationship between the load and each variable [2]. The complexity of the environment increases the difficulty of prediction. At the same time, the development of distributed power grids has increased the volatility of electricity, making load forecasting, especially the forecasting of 220KV substations, more difficult [3].

The operating characteristics of power generation equipment depend on the accuracy of the electrical load [4]. If the predicted load value is lower than the actual demand load value, there will be power shortages and power shortages, which will even affect the normal production of enterprises and people's lives, and cause unpredictable property losses to the people, society, and enterprises; if The predicted load value is higher than the actual demand power load, which will cause waste of resources, because electric energy has the characteristics of not being able to be stored in large quantities and used immediately. Therefore, power generation systems increasingly rely on accurate power load forecasting results. Therefore, continuously improving the accuracy of load forecasting is a requirement for the development of power systems [5].

2. Methodology

2.1. BP neural Network

Back Propagation (BP) based on error is a forward neural network with no feedback signal. The signal is transmitted in the order of input layer, hidden layer, and output layer [6]. The schematic diagram of the model is shown in Figure 1.

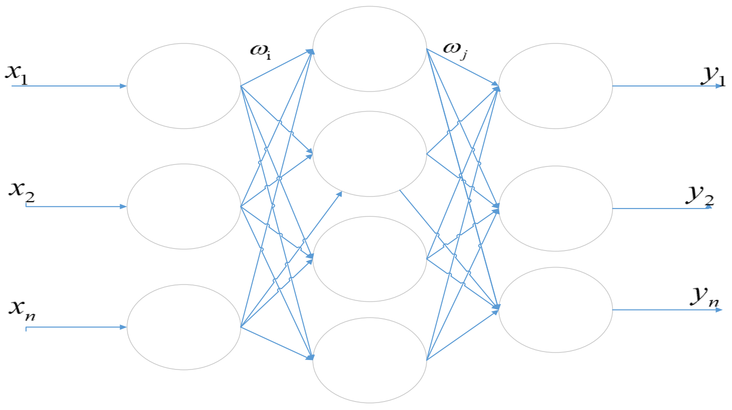


Figure 1. Schematic diagram of artificial neural network model

In figure 1, x_i represents the input variable, y_i represents the output variable, and ω_i is the weight of each input variable, which is used to represent the correlation between the two layers of neuron connections. The output of the upper layer neuron is weighted and input to the lower layer neuron. The BP neural network transmits signals through the forward propagation sequence, and the backward propagation error corrects the network parameters.

The signal transmission formula of N-dimensional variables from the input layer neuron to the j-th hidden layer neuron is as follows.

$$net_j = \sum_{i=1}^{n} w_{ij} x_i (i = 1, 2, \dots, n) \tag{1}$$

After calculating the activation function of the hidden layer, the output result of the hidden layer neuron Q_j is as follows [7].

$$Q_j = f(net_j + b_j) = \frac{1}{1+e^{-(net_j + b_j)}} (j = 1, 2, \dots, q) \tag{2}$$

In formula 2, b_j the threshold of the hidden layer neuron, and $f(*)$ is the activation function. Here, the Sigmiod function is used to map the input value to a continuous value from 0 to 1.

The transfer value from the hidden layer neuron j to the output layer neuron $k$is as follows.

$$net_k = \sum_{j=i}^{q} w_{jk} Q_j (j = 1, 2, \dots, q) \tag{3}$$

The output of neurons in the output layer Q_k is:

$$Q_i = f(net_i) \tag{4}$$

The error back propagation process of the BP neural network is to set the error value. If the error between the real value and the predicted value is greater, it will be fed back to the input layer to re-correct the weight and threshold. The error of a single sample is expressed E_p as:

$$E_p = \frac{1}{2}\sum_{i=1}^{n}(d_{pk} - Q_{pk})^2 \tag{5}$$

The cumulative error expression of the entire sample data set is:

$$E = \sum_{p=i}^{p} E_p = \frac{1}{2}\sum_{p=1}^{p}\sum_{k=1}^{n}(d_{pk} - Q_{pk})^2 \tag{6}$$

The BP neural network obtains the ideal output variable through forward propagation from front to back. If the result of the variable is not ideal, the value is continuously adjusted through reverse error propagation to reduce the error to the set value or the number of iterations [8].

2.2. PASA-PSO-BP Neural Network Model

2.2.1 PSO-BP Predictive Model

The PSO-BP model has the advantages of faster convergence and improved prediction accuracy. The basic idea of the PSO-BP model is: the position of each particle in the particle swarm is a multi-dimensional array, and the dimension of the array depends on the number of weights and thresholds, representing the set of weights in the current iteration process [9]. The weights and thresholds optimized by the PSO algorithm are used as the initial weights and thresholds of the neural network model, and the output error of the neural network is used as a measure of the fitness function value of the particles. The smaller the fitness value, the smaller the error. The higher the accuracy. In the solution space, the particles fly according to the speed formula and reach the new position where the fitness value drops, thereby determining the optimal weight and threshold.

The formula for particle velocity and position are [10]:

$$V_{id}^{k+1} = \omega V_{id}^{k} + C_1 random_1(0,1)(Pbest - X_{id}^{k}) + C_2 random_2(0,1)(gbest - X_{id}^{k}) \tag{7}$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \tag{8}$$

In formula 7 and 8, V_{id}^{k} and X_{id}^{k} represent the speed and position of the i -th particle, respectively, where k represents the current iteration number, $d$represents the dimension; ω is called the inertia factor, and ω is a non-negative number; C_1 and C_2 are called learning factors, Used to adjust the step length of learning. C_1 represents the individual learning ability of each particle, C_2 represents the social learning ability of the particle, in general, $C_1 = C_2 \in [0,4]$; $random(0,1)$ represents a random number between 0 and 1; $Pbest$ represents the highest individual in history. The fitness value, $Gbest$ represents the historical best fitness value in the particle swarm .

The formula of particle fitness function fit is as follows [11]:

$$fit = \frac{1}{N}\sum_{k=1}^{N}\sum_{t=1}^{m}(d_t - \hat{d}_t)^2 \tag{9}$$

In formula 9, N represents the training sample, d_t represents the actual output value of the neural network, and $\hat{d}_t$ represents the target output. When the particle swarm algorithm converges, the inertia weight plays an important role. When the weight is larger, the particles focus on the global search, and the convergence speed is slow; when the weight becomes small, the particles focus on the local search, which is easy to fall into the local optimum. In common particle swarm models, the size of ω is usually adjusted linearly or statically to obtain global or local optimization capabilities according to specific conditions. In this paper, the dynamic inertia weight formula is as follows:

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{k_{\max}} k \tag{10}$$

In formula 10, $k_{\max}$ represents the maximum number of iterations

2.2.2 Introduction to Simulated Annealing Algorithm

The particle swarm algorithm can be applied to many professional fields such as imaging and surveying, and has the advantages of fewer parameters and rapid iteration process. However, the particle swarm algorithm has the shortcoming of premature convergence in the early stage of the iteration, especially when dealing with the multi-peak search problem, it will fall into the local extremum.

The advantage of the simulated annealing algorithm is that it can retain some poor solutions with a certain probability in the early stage of the iteration, so as to avoid falling into the local extreme value prematurely. The flow of the simulated annealing algorithm is shown in Figure 2.

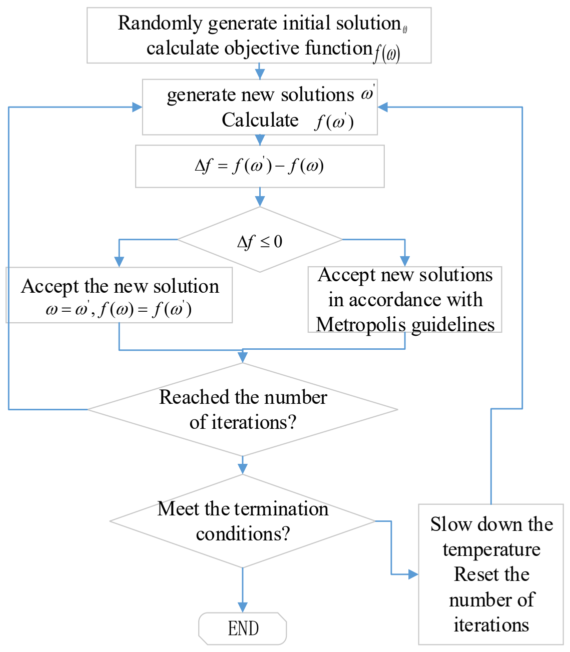

Randomly generate initial solution ω, calculate objective function $f(\omega)$
generate new solutions ω' Calculate $f(\omega')$
$\Delta f = f(\omega') - f(\omega)$
$\Delta f \le 0$
Accept the new solution $\omega = \omega', f(\omega) = f(\omega')$
Accept new solutions in accordance with Metropolis guidelines
Reached the number of iterations?
Meet the termination conditions?
Slow down the temperature Reset the number of iterations
END

Figure 2. Flow chart of simulated annealing algorithm

Simulated annealing is derived from the movement of molecules in physics. The larger the energy of the molecule, the more unstable it is. On the contrary, the smaller the energy of the molecule, the more stable it is. We assume that the initial state of the molecule is i, and its energy is E_i. After the movement, the new state is j, and the energy also becomes E_j. If the new energy is greater than the initial energy, $E_j > E_i$, keep this new state and mark it as an "important state", but if the energy of the initial state is greater than the energy of the new state, $E_i > E_j$, the new state will not be discarded at this time, but Judge through the given formula and retain the new state with a certain probability r.

$$\mathrm{r} = \exp(-\frac{E_j - E_i}{kt}) \tag{11}$$

In formula11, t represents temperature, k is Boltzmann's constant, and $\exp$ represents the natural index. Generate a random number $\xi \in [0,1)$, if ξ is less than the probability r, then the new state is retained, if ξ is less than the probability r, then the state is discarded.

The simulated annealing algorithm is a greedy algorithm, which has probabilistic jump characteristics. When performing a global search, it does not exclude all solutions with poor fitness, but retains some poorly fitness particles with a certain probability during iteration to the next iteration. The probabilistic jump feature of the simulated annealing algorithm can give the particles a chance to jump out of the local extreme when the particles fall into the local optimum, and improve the global convergence of the particles.

2.2.3 Optimizing PSO Algorithm Based on PASA

This paper proposes an improved particle swarm algorithm based on the optimized simulated annealing algorithm. The improved particle swarm algorithm can initialize particles uniformly during the initialization period and has a higher fitness value, and can ensure a global search in the early stage of the search; particles with poor fitness values are retained with a certain probability during the particle flight. Falling into the local optimum; at the same time, selecting particles with a high fitness value and uniform distribution at the beginning of the search speeds up the convergence speed and can ensure the globality of the food source judgment. The improved particle swarm algorithm steps are as follows:

Step 1: Build a BP neural network model, and initialize the number of network nodes and network layers;

Step 2: Initialize the particle swarm to generate a population of N particles, and set parameters for each particle;

(1) The total number of particle swarms N;

(2) The maximum number of iterations $\max Cycle$;

(3) The initial temperature T and annealing speed α of simulated annealing;

(4) Using a random initialization strategy to initialize N particles to generate N positions, that is, the initial solution $\mathrm{X}_i (i = 1, 2, ..., N)$;

(5) Use random initialization strategy to generate N-dimensional initial velocity value, $V_i^1 (i = 1, 2, ..., N)$;

(6) Initialize the inertia weight ω_t randomly.

(7) Assign each $Pbest_i$ value to the initial position of the current particle: $Pbest_i = X_i^1$.

(8) Select the optimal value of fitness value from the current initial population and set it to the current global optimal value $Gbest$.

Step 3: Calculate the new particle position X_i^{t+1} according to the current particle position X_i^t and velocity V_i^t, which is the candidate solution for the $t+1$ th iteration;

Step 4: Calculate the fitness function value E_i^{t+1} of the current particle i at the new position $t+1$; if E_i^{t+1} is greater than $Pbest_i$, then set $Pbest_i = E_i^{t+1}$.

This article uses the following fitness function:

$$E_i = \begin{cases} 1/(1+E_i) & \mathrm{E_i} \geq 0 \\ 1+\mathrm{abs}(\mathrm{E_i}) & \mathrm{E_i}<0 \end{cases} \tag{12}$$

Step 5: Calculate the current best fitness value of each particle $Pbest$ and ,compare it with the current global extreme value $Gbest$. If the best fitness value is greater than $Gbest$, Then we will set the current best fitness value of the particle to $Gbest$: $Gbest = \max(pbest_1, pbest_2, ..., pbest_N)$.

Step 6: Update the flying speed of the particles. This paper proposes an adaptive inertia weight based on fitness value. ω is the weight of inertia in the formula, which is used to adjust the memory item of flying particles. The larger ω is, the global search ability is stronger, and the smaller ω is, the local search ability is stronger. This paper dynamically adjusts the inertia weight based on the change of the fitness value, and dynamically and adaptively solves the local extremum and early convergence problems of particles.

$$\omega_i^{t+1} = \begin{cases} \omega_i^t(\exp(\frac{1}{\Delta E+1}-1)+1) & \Delta E > 0 \\ \omega_i^t(\exp(\frac{1}{\Delta E+1}-1)) & \Delta \mathrm{E} \leq 0 \end{cases} \tag{13}$$

In formula 13, $i \in \{1,2,...,N\}$, $t \leq \max Cycle$. The adaptive inertia weight based on the fitness value can improve the diversity and globality of the population, and changes have been made to solve the problem of premature convergence of particles and falling into local extremes.

Step 7: After obtaining the velocity V_i^{t+1} of the particle, obtain the new position X_i^{t+1} of the particle through the formula.

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{14}$$

Step 8: Calculate the fitness value of the new position of the particle, and compare the value with the fitness value of the old position. This paper proposes an improved simulated annealing algorithm to determine whether to enter the new particle position.

(1) Calculate the fitness difference between the new position of the particle and the old position $\Delta \mathrm{E}_i = \mathrm{E}_i^{t+1} - \mathrm{E}_i^t$;

(2) If the diversity of the particle swarm is low, it is easy to converge locally. This paper proposes a method based on the variance of the Euclidean distance to indicate the degree of aggregation of particles and other particles, and uses the variance of the Euclidean distance of the particles to indicate the uniformity of the particle distribution of the population, and uses this value as a feedback value to adjust the population of particles in simulated annealing Diversity.

$$D(t) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i(t) - \overline{d(t)})^2} \tag{15}$$

In formula 15, $\mathrm{d}_i(t)$ is the Euclidean distance between the particle at its current position and other particles in the population, and $\overline{\mathrm{d}(t)}$ z represents the average Euclidean distance of all particles at the current position. $D(t)$ represents the variance of the distance of the particles in the population from the current position, and represents the uniformity of the particle distribution in the solution space. The larger the variance, the uneven distribution of the optimal solution of the particle distance; the smaller the variance, the relatively uniform distribution of the optimal solution of the particle distance.

When $\Delta E \le 0$, the fitness value decreases, the particles are closer to the optimal solution, and the new particle position is accepted at this time;

When $\Delta E > 0$, the fitness value becomes larger, indicating that the particle principle is the optimal solution, but if it satisfies

$$\exp(-\frac{\Delta E}{T} \bullet \frac{1}{D(t)+1}) > rand(0,1) \tag{16}$$

Where $rand(0,1)$ represents a random number between 0 and 1, and the new particle position is also accepted; otherwise, the new position is rejected $X_i^{t+1} = X_i^t$.

(3) If the new particle position is accepted, the annealing formula $T_{i+1} = \alpha T_i$ is executed.

Step 9: Determine whether the termination condition is met. If the maximum number of iterations is not reached or the accuracy is met, then go to step 3; if the condition is met, the global optimal solution is output.

3. Analysis of Algorithms

3.1. Test Function Performance Analysis

In order to verify the performance of the PASA-PSO algorithm, this paper selects the following standard test functions to test the algorithm: Shubert, Ackley, Griewank, Rastrigin, Rosenbrock and Schaffer. Test and compare the PASA-PSO proposed in this paper and the traditional PSO algorithm.

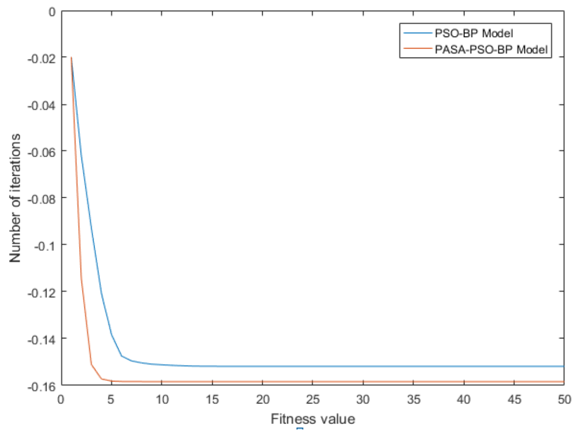
The test function parameter settings are shown in Table 1.

Table 1. Test function

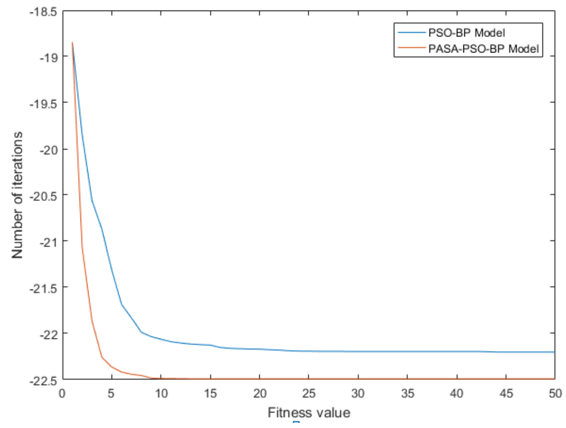
Test function	Function	range	Optimal solution
Shubert	$f_1(x) = 418.9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500,500]^D$	0
Ackley	$f_2(x) = -a\exp(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - \exp(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)) + a + \exp(1)$	$[-32,32]^D$	0
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-10,10]^D$	0
Rastrigin	$f_4(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12,5.12]^D$	0
Rosenbrock	$f_5(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	$[-30,30]^D$	0
Schaffer	$f_6(x) = 0.5 + \frac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1 + 0.01(x_1^2 + x_2^2))^2}$	$[-10,10]^D$	0

Before conducting the experiment, set the parameters for the PASA-PSO and SA-PSO models. Set the population size to 20, and set the maximum number of iterations of the test function $\text{maxCycle}=800$, $T = 1000^{d}C$, $\alpha = 0.9$. The simulation laboratory runs on a 64-bit windows10 computer and runs 50 times continuously on Matlab.

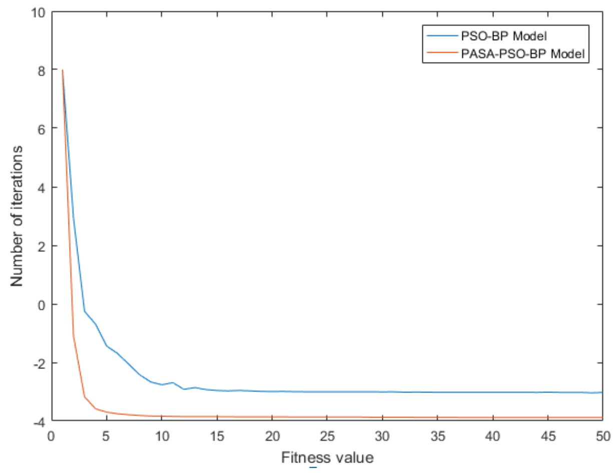
The experimental results are shown in Figure 3.



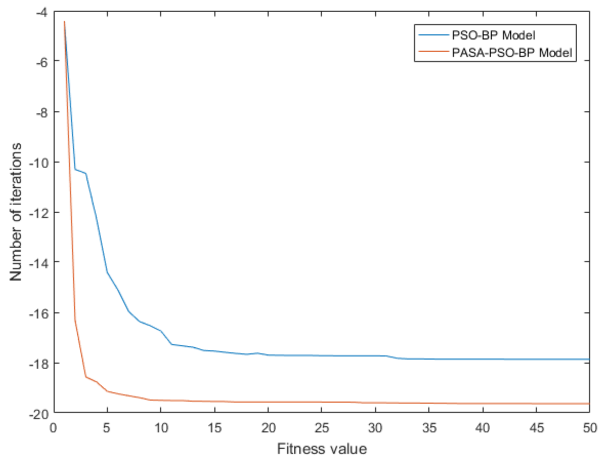
a) ShubertFunction



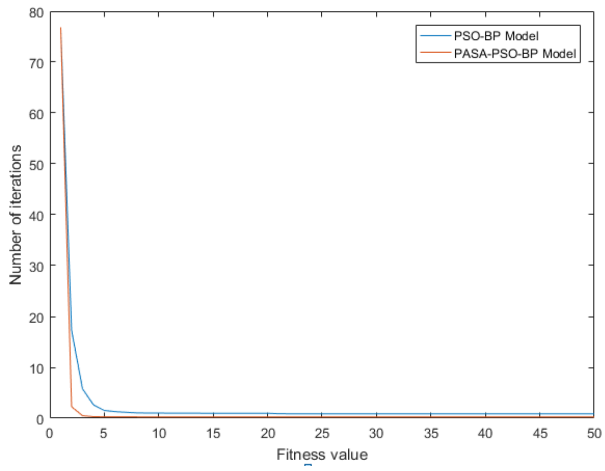
b) AckleyFunction



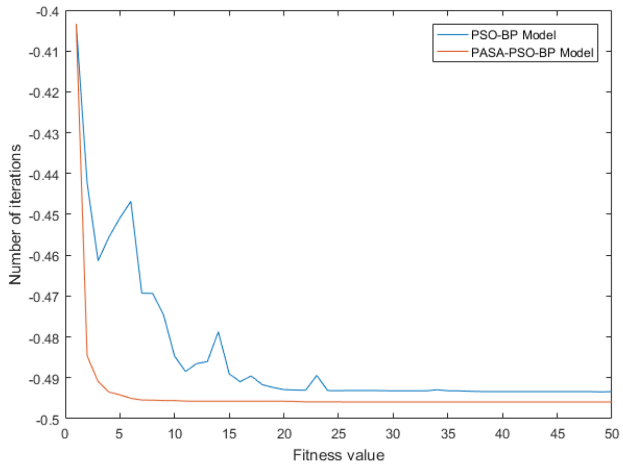
c) GrieWankFunction



d) RasringinFunction



e) RosenbrockFunction



f) SchafferFunction

Figure 3. Test results of six test functions

The test results in Figure 3 show that the convergence accuracy of the PASA-PSO algorithm has been improved compared to the PSO-BP algorithm. Especially in the tests of the Ackley function and the Schaffer function, the PASA-PSO algorithm has better convergence performance. In the Rosenbrock function and Shubert test results, the accuracy convergence effect of PASA-PSO is not significant. Comparing the test curve in Figure f), PASA-PSO has a faster convergence rate, but the convergence rate in other test functions is not significantly improved.

Establish PASA-PSO-BP model proposed in this paper and traditional PSO-BP model. Then comparing the two forecasting models, the true value and predicted value curve of the load are shown in Figure 4.

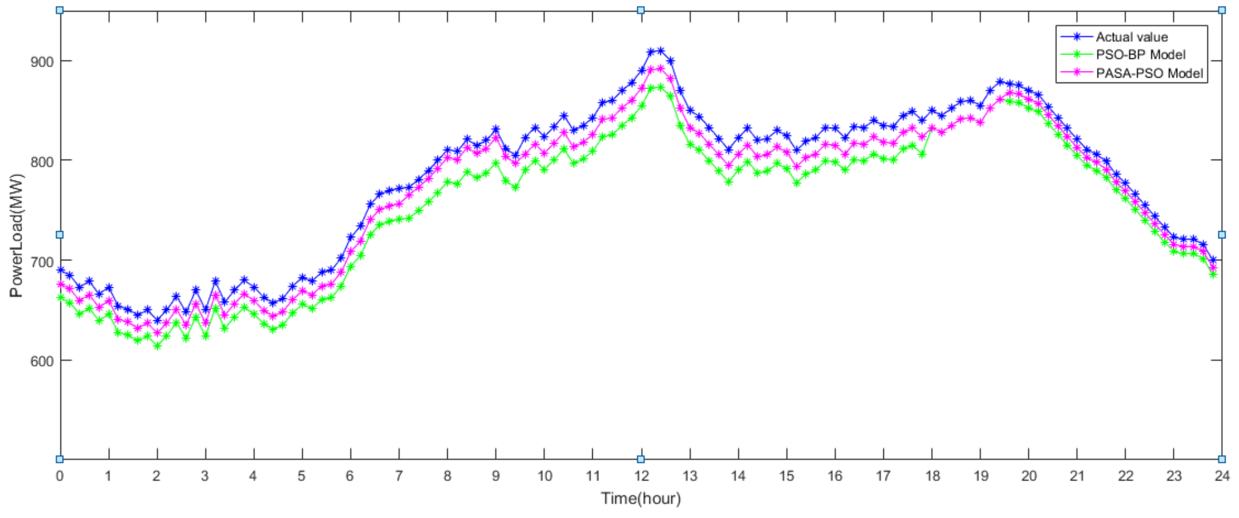


Figure 4. Comparison of the prediction results of the two models

The above figure is a comparison between the PSO prediction model and the PASA-PSO model. The relative error comparison figure of the prediction results is shown in Figure 5.

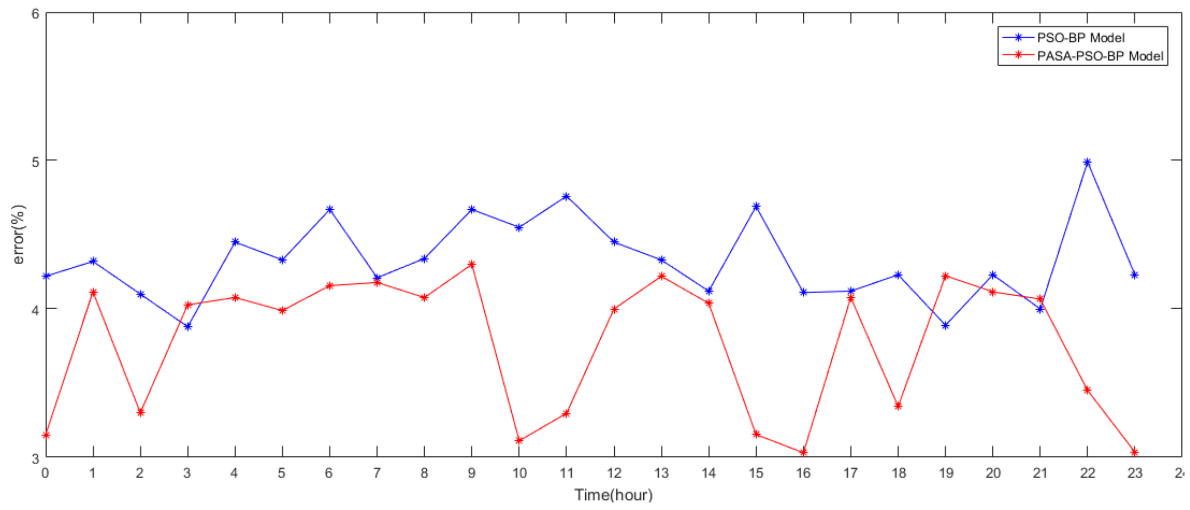


Figure 5. Comparison of prediction results error

From figure 5, we can see that PASA-PSO has a better load fitting effect, with an average relative error of 3.775%. The average relative error of the PSO model is 4.335%. It shows that the improved particle swarm algorithm greatly improves the prediction accuracy and reduces the relative error.

4. Conclusion

This paper establishes a BP neural network model to forecast the load. The PASA-PSO-BP neural network model is proposed, combined with the improved simulated annealing algorithm to optimize the PSO algorithm, and the BP neural network model is established, which greatly improves the convergence speed and accuracy of the network. Taking the same set of data as an example, comparing the prediction effects of the PSO-BP neural network and the PASA-PSO-BP neural network model, the PASA-PSO-BP neural network model has the highest prediction accuracy, indicating that the PASA-PSO algorithm is used for BP The improvement effect of neural network is remarkable.

This paper proposes a variance representation of population diversity (PA) based on Euclidean distance, and uses the probabilistic jump feature of the simulated annealing algorithm to propose a simulated annealing (PASA) algorithm based on population diversity. After matlab test function test, PASA-PSO algorithm has higher convergence accuracy. By predicting the same set of data, the prediction accuracy is improved from the average relative error of the PSO model from 4.335% to 3.775%.

References

[1] Cheng Yuye. Research on short-term power load forecasting based on artificial neural network. Zhejiang University, 2017.

[2] Wang B, Tai N L, Zhai H Q, et al.A new ARMAX model based on evolutionary algorithm and particle swarm optimization for short-term load forecasting [J]. Electric Power Systems Research, 2008, 78(10):1679-1685.

[3] Huang S J, Shih K R. Short-term load forecasting via ARMA model identification including non-Gaussian process considerations[J]. Power Systems, IEEE Transactions on, 2003.

[4] Tang Junjie, Niu Huanna, Yang Minghao. Periodic autoregressive short-term load forecasting based on linear correlation analysis [J]. Power System Protection and Control, 2010, 038(014): 128-133.

[5] Ye Zhou, Huang Ting, Dai Ren, et al. Seasonal autoregressive moving average hybrid model and its application in power load forecasting [J]. Sichuan Electric Power Technology, 2001, 24(1).

[6] Mandal P, Senjyu T, Urasaki N, et al. A neural network based several-hour-ahead electric load forecasting using similar days approach[J]. INTERNATIONAL JO- URNAL OF ELECTRICAL POWER AND ENERGY SYSTEMS, 2006.

[7] Bashir Z A, El-Hawary M E. Applying Wavelets to Short-Term Load Forecasting Using PSO-Based Neural Networks[J]. IEEE Transactions on Power Systems,200 9, 24(1):p.20-27.

[8] Lauret P, Fock E, Bayesian neural network approach to short time load forecasting[J].Energy conversion & Management,2008,49(5):1156-1166.

[9] Wang Wenqing. Research on power system short-term load forecasting based on convolutional neural network [D]. Qingdao University, 2020.

[10] Fang T, Lahdelma R. Evaluation of a multiple linear regression model and SARIMA model in forecasting heat demand for district heating system[J]. Applied En- ergy, 2016.

[11] Li S, Goel L, Wang P. An ensemble approach for short-term load forecasting by extreme learning machine[J]. Applied Energy, 2016, 170(may 15):22-29.

[12] Lou C W, Bong M C. A novel random fuzzy neural networks for tackling uncertainties of electric load forecasting[J]. International Journal of Electrical Power & Energy Systems, 2015, 73(dec.):34-44.

[13] Slama, JB Hadj, Lahouar, et al. Day-ahead load forecast using random forest and expert input selection[J]. Energy conversion & management, 2015.

[14] Shi Delin. Research and implementation of power load forecasting based on neural network [D]. Shandong University.