ATLANTIS PRESS
Advances in Engineering Research (AER), volume 132
2017 Global Conference on Mechanics and Civil Engineering (GCMCE 2017)

A Software Reliability Prediction Algorithm Based on MHPSO - BP Neural Network

Dong Xu [1,a], Shaopei Ji [1,b], Yulong Meng [1,c,*] and Ziying Zhang[1,d]

[1] College of Computer Science and Technology, Harbin Engineering University, Harbin, China

[a]xudong@hrbeu.edu.cn, [b]jishaopei@hrbeu.edu.cn,

[c]mengyulong@hrbeu.edu.cn, [d]zhangziying@hrbeu.edu.cn

*corresponding author

Keywords: Software reliability prediction, Attractor, MHPSO, BP neural network.

Abstract. Because the weights and thresholds of BP neural network usually adopt random assignment, there is a problem of low accuracy in software reliability prediction. In order to solve this problem, a software reliability prediction algorithm (MHPSO-BP) based on multi-layer heterogeneous PSO optimized BP neural network is proposed in this paper. In this algorithm, the population structure of the particle swarm is set to the hierarchical structure, and the velocity updating equation of the particle is improved by using the attractor. The information interaction between the particles is enhanced, and the optimization performance of the particle swarm optimization algorithm is improved. And then use the improved PSO to optimize the weight and threshold of the BP neural network. The software reliability prediction experiment was performed using the JM1 software defect data set of the NASA-MDP project during the experiment. The results show that the proposed method has better predictive performance than the traditional BP neural network.

1. Introduction

Software reliability is one of the most important indicators to judge the quality of software products. How to model and evaluate software reliability accurately is one of the most important problems in software reliability research [1]. The method of establishing software reliability model can be divided into two categories: the method based on statistical analysis and the method based on data-driven [2]. The model established by the former requires the initial hypothesis of the software failure process and the internal error elimination process. Then, we use the mathematical mechanism of stochastic process to model the software. Finally, we use the statistical method of data statistics to analyze the reliability of software. The normal models are NHPP model [3], L-W model [4] and J-M model [5]. In practice, these assumptions and the actual situations have difference of different degrees, so the applicability and accuracy of prediction are limited in the actual software reliability assessment and prediction. The model of the second method is based on a method of software failure data, which is the main research feature because of the absence of assumptions and high applicability and prediction accuracy. At present, the data-based software reliability model is based on the SVM [6], the gray prediction theory [7], the hybrid forecasting model based on multiple forecasting methods [8-9] and the artificial neural network model [10], etc.

Because of its strong adaptive and self-learning ability, BP neural network can obtain a relatively ideal forecasting model if it is given only enough data when it is training which makes it a new approach of software reliability prediction. But there are some shortcomings: First, before the network model training, the random initialization of BP network connection weights and thresholds makes the network easy to fall into the local extreme points, which affects the nonlinear learning ability, that is, the accuracy of prediction. Second is how to determine the neural network structure, that is, there is not the exact formula of the number of hidden layer nodes. Then the improper selection will lead to the over-fitting or lack of learning ability [11-13], which affect the generalization of the network capacity.

There are two main methods to optimize the BP neural network: 1) Improve the BP algorithm by improving the related parameters, such as the improvement of the error function and the improvement of the excitation function. 2) Optimize the BP neural network based on the group intelligence algorithm, such as BP neural network optimization based on GA [14], optimized BP neural network based on PSO algorithm [15]. On the basis of this, this paper proposes a software reliability prediction method based on MHPSO-BP neural network. By optimizing the particle swarm group structure and attracting particle concept, the information interaction between particles is enhanced, and the optimization performance of PSO algorithm is improved. When it is used to optimize the parameters and threshold of BP neural network, it makes the prediction performance of BP neural network be improved.

2. Particle Swarm Optimization for BP Neural Network

2.1 Particle Swarm Optimization and Relevant Variants

Particle swarm optimization (PSO) is an intelligent optimization algorithm for simulating the foraging of birds proposed by Kennedy and Eberhart in 1995. Each particle in the search space adjusts their own speed and position to optimize based on Eq.1 and Eq.2, until the termination of the conditions to meet the convergence.

$$\mathrm{V}_{i,j}^{t+1} = \omega \mathrm{V}_{i,j}^{t} + c_1 r_{1,i,j}^{t}\left(\hat{y}_j^t - x_{i,j}^t\right) + c_2 r_{2,i,j}^{t}\left(y_{i,j}^t - x_{i,j}^t\right). \tag{1}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + \mathrm{V}_{i,j}^{t+1}. \tag{2}$$

Here, $V_{i,j}^t$ denotes the velocity value of particle i at time t, $x_{i,j}^t$ is the position of the particle i at time t, ω is a parameter called inertia weight representing how much particles' memory can influence the new position, c_1 and c_2 are two constant acceleration coefficients, $y_{i,j}^t$ is the personal best solution of particle i at time t, and $\hat{y}_j^t$ is the global best solution known at time t; $r_{1,i,j}^t$、$r_{2,i,j}^t$ are two random numbers.

2.2 Proposed Algorithms

As the particles in the PSO gather to their local optimal position and the global optimal location of aggregation which forms the rapid convergence effect of the particle population, it is prone to fall into the local extreme, premature convergence or stop phenomenon [16].So, a multi-layered heterogeneous dynamic particle swarm optimization (MHPSO) algorithm is proposed, which focuses on establishing the vertical interaction between multiple layers. The particle population structure is shown in Fig.1, where each layer contains the same number of particles.

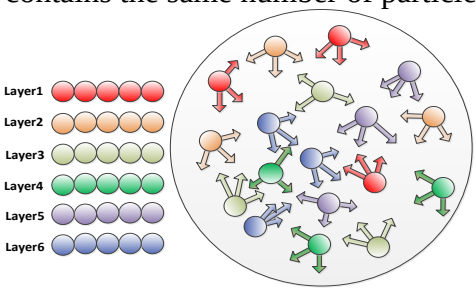

Fig.1 The structure chart of particle population

At each iteration, the particles are sorted according to their current fitness values and are assigned to different levels in turn. The smaller the fitness values of the particles, the higher the level of the particles. In the algorithm, the particles are attracted by the particles in the direct upper layer, and the particles in the direct upper layer are its attracting particles, which are also the attracting particles for their lower layers. The particles in the top layer can only be attracted by other particles of the same layer. For example, if the particle is at the third level, the particles in the fourth layer are the particles

that attract the particles at the third level, and the particles in the second layer attract particles at the third level, too. However, the particles in the uppermost layer are exceptions: their attracting particles are other particles in the same layer, mainly because they all have relatively optimal fitness values in the population. In addition to moving to their individual optimal position and global optimal position during the movement, the particles move to the position where they attract the particles.
In the new algorithm, the speed update formula (Eq.3) adds an additional term from the attracting particles:

$$V_{i,j}^{t+1} = \omega V_{i,j}^{t} + c_1 r_{1,i,j}^{t}\left(\hat{y}_j^t - x_{i,j}^t\right) + c_2 r_{2,i,j}^{t}\left(y_{i,j}^t - x_{i,j}^t\right) + \sum_{a=1}^{A_j^i} c_3 r(i)_{a,j}^t\left(x(i)_{a,j}^t - x_{i,j}^t\right) \quad (3)$$

Here, $x(i)_{a,j}^t$ is the position where the attracting particle a of particle i is located, A_j^i is the total number of attractive particles of i, and c_3 is the constant acceleration coefficient, $r(i)_{a,j}^t$ is the attracting coefficient corresponding to the attracting particle a of i.
In order to ensure that the influences which the particle affected by its attractor is balanced, to improve the robustness of the algorithm, the particle attracting coefficient is divided into the following two cases:

1) When the value of $S(i)_{a,j}^t$ is less than $\bar{S}(i)_j^t$,

$$r(i)_{a,j}^t = r_{min}^t + \frac{(r_{max}^t - r_{min}^t)\left(S(i)_{a,j}^t - S(i)_{min,j}^t\right)}{\bar{S}(i)_j^t - S(i)_{min,j}^t}. \quad (4)$$

2) When the value of $S(i)_{a,j}^t$ is greater than $\bar{S}(i)_j^t$,

$$r(i)_{a,j}^t = r_{max}^t - \frac{(r_{max}^t - r_{min}^t)\left(S(i)_{max,j}^t - S(i)_{a,j}^t\right)}{S(i)_{max,j}^t - \bar{S}(i)_j^t}. \quad (5)$$

Here, r_{min}^t and r_{max}^t are the minimum and maximum values of the attracting coefficient. $S(i)_{min,j}^t$ and $S(i)_{max,j}^t$ are the minimum and maximum values of the particle position of particle i respectively. $S(i)_{a,j}^t$ represents the distance from the attractor a of particle i to particle i. $\bar{S}(i)_j^t$ is the average of all the attractive particles positions of particle i. When the value of $S(i)_{a,j}^t$ is less than $\bar{S}(i)_j^t$, the attractor a will correspond to a larger attraction factor; when the value of $S(i)_{a,j}^t$ is greater than $\bar{S}(i)_j^t$, the attractor a will correspond to a smaller attraction factor.

3. BP neural network based on MHPSO optimization

The traditional BP network uses a gradient descent method that makes the weight converge to a certain value, but it cannot be guaranteed that it is the global minimum of the error plane. The modified PSO is used to replace the gradient descent method in the algorithm of MHPSO optimization BP neural network.This algorithm is to optimize the BP neural network connection weight and threshold by PSO algorithm, which can improve the performance of BP algorithm. The algorithm steps are as follows:
Step1: Initialize the particle swarm parameters and the structure of BP network. Particle swarm parameters include population size, population stratum, number of iterations, learning factors, and limited range of particle position and velocity. Among them, the initial value of the particle position and velocity is a random value. The structure of the initial BP neural network mainly includes the number of neurons in each layer of the neural network and the number of layers of the hidden layer.
Step2: Determine the evaluation functions of the particles. The fitness function of the particles in the population is defined as Eq.6:

$$fit_i = \sum (Y_{ij} - y_{ij})^2, i = 1,2,\dots,n \quad (6)$$

Where n is the population size, Y_{ij} is the sample output value, and y_{ij} is the actual output value.

Step3: Calculate the fitness value of each particle and construct the population hierarchical structure. The fitness values of each particle are calculated and sorted based on Eq.6, and then build population hierarchical structure.
Step4: Update the local optimal position of the particle and the global optimal position.
Step5: Update the speed and position of the particle itself based on Eq.2、Eq.3、Eq.4 and Eq.5.
Step6: If the end condition of the iteration (good enough position or maximum number of iterations) is reached, then turn to the step3 and continue to iterate.
Step7: The obtained optimal particles are assigned to the connection weights and thresholds of the BP neural network. After the BP neural network prediction model is trained, the optimal time is predicted by the output time series.

4. Experiment Analyses

4.1 Experiment I

In order to evaluate the performance of MHPSO, PSO and QPSO were selected as the comparison algorithm in the experiment I, and the functions Ackley and Rastrigin were selected as the reference function. The parameter values used in the three algorithms are listed in Tab.1.

Table 1. Parameters setting rules

parameter	PSO	QPSO	MHPSO
ω	linearly decreasing from 0.7 to 0.2	linearly decreasing from 0.7 to 0.2	linearly decreasing from 0.7 to 0.2
c_1	linearly decreasing from 2.5 to 0.5	linearly decreasing from 2.5 to 0.5	linearly decreasing from 2.5 to 0.5
c_2	linearly increasing from 0.5 to 2.5	linearly increasing from 0.5 to 2.5	linearly increasing from 0.5 to 2.5
c_3	null	null	the number of attractors divided by 2.5
r_1	obey uniform distributionU(0,1)	obey uniform distributionU(0,1)	obey uniform distributionU(0,1)
r_2	obey uniform distributionU(0,1)	obey uniform distributionU(0,1)	obey uniform distributionU(0,1)
r_{min}^t	null	null	0.3
r_{max}^t	null	null	1.2

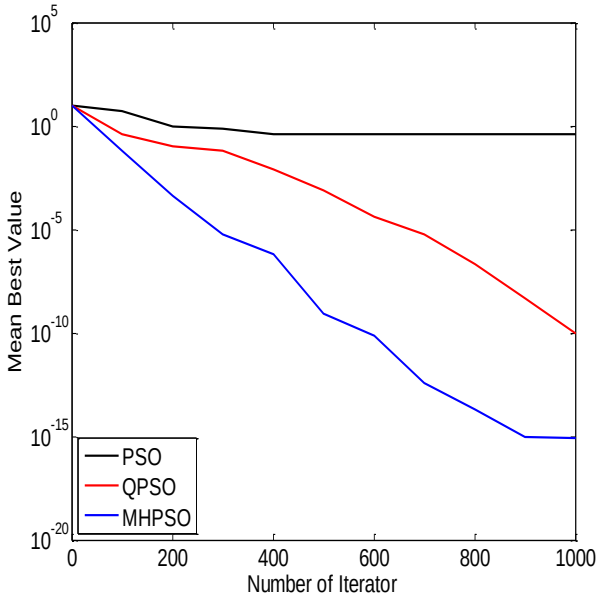

Fig. 2. The optimization results of Ackley function

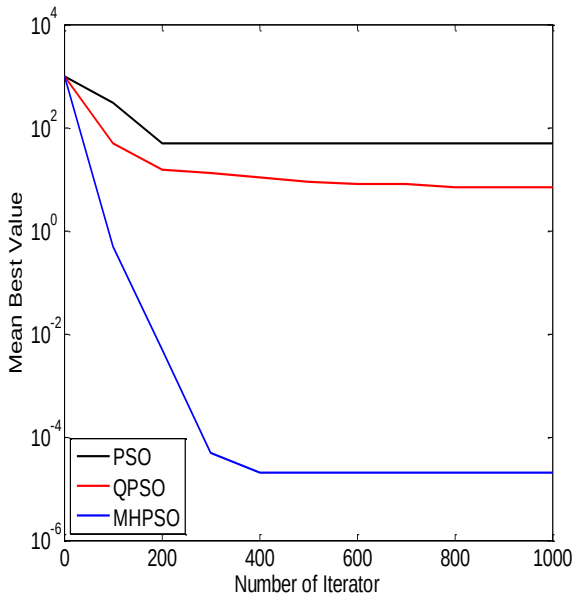

Fig. 3. The optimization results of Rastrigin function

Fig.2 and Fig.3 show the optimized and compared results of the three algorithms for Ackley and Rastrigin. As can be seen from the two figures, from the initial stage, MHPSO reflect the excellent search capabilities. When the multi-peak function is tested, it achieves the required accuracy in a very short period of time. When the unimodal function is tested, the optimal value of MHPSO show a linear decrease trend, and the exploration ability is very strong.

As the experiment shows, the MHPSO algorithm shows better search efficiency than the other two algorithms, and the optimization results for the benchmark function are more obvious. The interaction between particles in MHPSO algorithm is more effective, which reduces the possibility of population falling into local extremum and also improves the local development capability of population in the feasible domain space and the convergence speed of the algorithm.

4.2 Experiment II

In order to demonstrate the effectiveness of MHPSO-BP neural network prediction algorithm (BPPSO-BP), BP neural network prediction algorithm (BP) and PSO-BP neural network prediction algorithm (PSO-BP) were used to be the compared experiments.

In the experiment II, JM1 from the NASA-MDP project was selected as the experimental comparison data set. JM1 data set has 17816 modules. Each module has 22 attributes, of which 21 for the characteristic attribute, 1 for the target attribute. In the JM1, defective module has 2102 and no defective module has 8776. Because of the large amount of dirty data in the data set, this paper deletes the dirty data of the JM1 dataset. Finally, 4000 data are randomly selected as experimental samples, which 30% of the defective samples and 70% of the defective samples.

1) Data dimensionality reduction

In order to eliminate the correlation between attributes and reduce the dimension of input data, the paper introduces the principal component analysis method to extract the feature of the sample data. The eigenvalues of the principal components, contribution rates and cumulative contribution rates of each principal component are shown in Tab.2 (only 7 main components are listed here).

Table 2. The table of main component statistics

main ingredient	eigenvalues	contribution rate（%）	cumulative contribution rate（%）
P1	28.1132	73.98	73.98
P2	3.3542	8.83	82.81
P3	1.9381	5.10	87.91
P4	1.3829	3.64	91.55
P5	1.0379	2.73	94.28
P6	0.5743	1.51	95.79
P7	0.2457	0.65	96.44

2) Parameters setting

The experiment adopts three-layer BP neural network structure, and sets the number of input layer nodes to 7, the number of hidden layers to 1, the number of hidden nodes initialized to 12, and the number of output nodes to 1. The parameter settings of PSO and MHPSO are the same as Table 1.

3) Analysis of results

In the experiment, the following categories of behaviors are defined, where the correct prediction is positive (True Positive, *TP*), which indicates that the data that is actually defective during the classification process is correctly classified as defective; the error prediction is negative (False Negative, *FN*) indicates that the data that is actually defective is predicted to be defect-free; the error prediction is positive (False Positive, *FP*) indicates that the data that is actually defect-free is predicted to be defective; the correct prediction is negative (True Negative, *TN*) is defined as a non-defective data prediction correctly predicted as defect-free data. In order to measure the validity of the algorithm comprehensively, the four indexes of recall rate, precision rate, accuracy rate and Fmeasure value are used to measure the effectiveness of the algorithm.

The accuracy rate is used to measure the proportion of all correctly classified samples to the total sample, the formula is : $(TP+TN)/(TP+FN+FP+TN)$.

The precision rate is used to measure the accuracy of detecting defective samples, the formula is : *TP/(TP+FP)*.
The recall rate reflects the proportion of defective samples that are correctly determined to the total defect, the formula is : *TP/(TP+FN)*.
The Fmeasure value represents the harmonic average of the accuracy rate and recall rate, the formula is : $2/(\frac{1}{accuracy\ rate}+\frac{1}{recall\ rate})$.

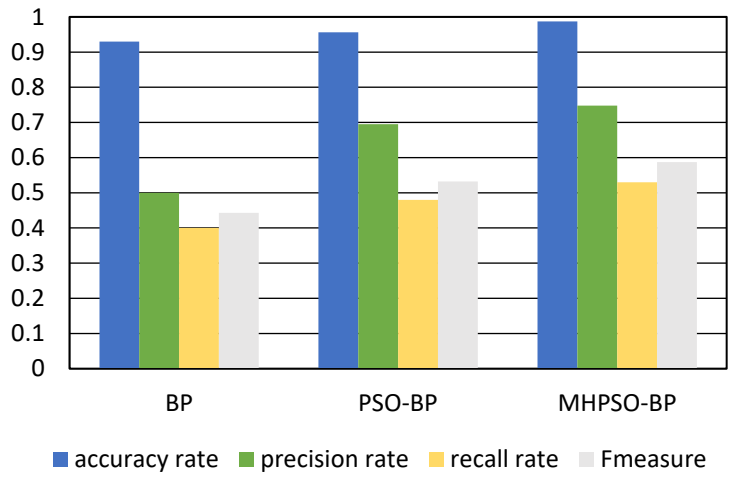

Fig. 4. The comparison of the predicted results

The comparison of the predicted results of the three models are shown in Fig.4. It can be seen that MHPSO-BP enhances the ability of information interaction between particles, because it avoids the rapid convergence of the population in the PSO where the particles are clustered toward the best position of their own history or the best position of the population history. The prediction accuracy is higher than PSO-BP and BP, indicating that MHPSO-BP is effective for the software reliability prediction.

5. Summary

Aiming at the problem that BP neural network is used to predict local minimum defects and slow convergence rate, a multi-layered heterogeneous dynamic particle swarm optimization algorithm --- MHPSO is proposed to optimize the weights and thresholds of BP neural networks, and the software reliability prediction experiment based on software defect data set is used. Compared with PSO-BP and BP, this algorithm improves the convergence speed and has better prediction accuracy.

Acknowledgement

This research was financially supported by the National Natural Science Foundation of China (Grant NO. 61502118).

References

[1] John B, Kadadevaramath R S, and Edinbarough I A, A Brief Review of Software Reliability Prediction Models, *International Journal for Research in Applied Science & Engineering Technology,* vol. 5, pp. 991-996, 2017.

[2] Aggarwal G, and Gupta V K, Software Reliability Growth Model, I*nternational Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, pp. 475-479, 2014.

[3] Ravishanker N, Liu Z, and Ray B K, NHPP models with Markov switching for software reliability[J]. *Computational Statistics & Data Analysis*, vol. 52, pp. 3988-3999, 2008.

[4] Pandey A K, and Goyal N K, Early Software Reliability Prediction. *Studies in Fuzziness & Soft Computing*, vol. 30, pp. 210–220, 2013.

[5] Yamada S, Software reliability modeling: fundamentals and applications. *Springer Science & Business Media*, 2013.

[6] Jin C, and Jin S W, Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms, *Applied Soft Computing*, vol. 15, pp. 113-120, 2014.

[7] Jin A, Jiang J H, and Lou J G, Software reliability modeling based on grey system theory, *Journal of Computer Applications*, vol. 29, pp. 690-694, 2009.

[8] Lo J H, A data-driven model for software reliability prediction, *IEEE International Conference on Granular Computing*, vol. 29, pp. 326-331, 2012.

[9] Xiao X, and Dohi T, Wavelet Shrinkage Estimation for Non-Homogeneous Poisson Process Based Software Reliability Models[J], *IEEE Transactions on Reliability*, vol. 62, pp. 211-225, 2013.

[10] Hu Q P, Xie M, and Ng S H, Software Reliability Predictions using Artificial Neural Networks, *Computational Intelligence in Reliability Engineering*, vol. 40, pp. 197-222,2007.

[11] Ru-Ping L I, and Zhu L, BP Neural Network Algorithm Improvement and Application Research, *Journal of Heze University*, 2016.

[12] Yang J, and Huang L. An Improvement and Application of Genetic BP Neural Network, *International Conference on Computational Intelligence and Security*, pp. 10-13, 2016.

[13] L. H. Jia, X. R. and Zhang, Analysis and Improvements of BP Algorithm, *Computer Technology and Development*, vol. 16, pp. 101-103, 2006.

[14] Liu L, and Jiang Z, Research on software reliability evaluation technology based on BP neural network, *Computer and Information Science, 2016 IEEE/ACIS 15th International Conference on. IEEE*, pp. 1-4, 2016.

[15] Ch S, and Mathur S, Particle swarm optimization trained neural network for aquifer parameter estimation. *KSCE Journal of Civil Engineering*, vol. 16, pp. 298-307, 2012.

[16] Clerc M, Particle Swarm Optimization// Fractional Order Darwinian Particle Swarm Optimization, *Springer International Publishing*, pp. 149-150, 2016.