IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 23, 2019, accepted February 15, 2019, date of publication March 5, 2019, date of current version March 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2903081

Integrating Artificial Bee Colony Algorithm and BP Neural Network for Software Aging Prediction in IoT Environment

JING LIU, (Member, IEEE), AND LINGZE MENG
College of Computer Science, Inner Mongolia University, Hohhot 010021, China
Corresponding author: Jing Liu (liujing@imu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61662051.

ABSTRACT Software aging is a common phenomenon that exists in systems that require long periods of operation, especially in Internet-of-Things environments. The back propagation (BP) neural network has been adopted widely to predict the trend of software aging. However, the weight and threshold of the BP neural network are randomly initialized, so it is easy to get the unsatisfactory local optimal solutions and the convergence speed of computing is slow. In this paper, we propose a novel software aging prediction method using the artificial bee colony algorithm to optimize the BP neural network model for achieving better software aging prediction accuracy. The experiment results show that our method fits the prediction trend of software aging more accurately than the traditional BP neural network, and our method also has faster convergence speed and more stable prediction results.

INDEX TERMS Artificial bee colony algorithm, BP neural network, software aging, prediction accuracy.

I. INTRODUCTION

Developing the intelligent and collaborative data processing software system for Internet of Thing (IoT) environments is quite significant. With the increasing complexity of such software systems, the frequency of system failures is also increasing. There are three main causes of system failure, i.e., hardware failure, software failure, and human design failure. Software failure is the major cause and mostly concentrated on software aging. Software aging refers to software performance degeneration caused by memory leakage, unreleased file locks, untimely data updates, storage space fragmentation, and accumulation of rounding errors during long-term uninterrupted operation of the software. These factors eventually lead to software failure [1]. The occurrence of software aging not only reduces the reliability of the system, but also endangers the safety of people's lives and property. So solving the software aging problem is crucial to the availability and reliability of the software system in IoT environments. In this field, how to accurately predicting the software aging time is an important and indispensable research issue, which could help to improve the software running reliability for IoT application platform.

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

At present, the analysis methods for software aging prediction are mainly divided into two categories. One is the model based analysis and the other one is the measurement based analysis. Model based analysis builds a state transition model based on different states of the system. The models are generally constructed using a Markov chain or a random Petri net [2]–[5], and the analysis of this model determines the time of rejuvenation. The rejuvenation often adopt the restarting the software system. While, measurement based analysis first collects the running states of system resources, such as CPU, memory, and so on. The collected data are then processed using time series analysis or machine learning algorithms, and finally the exact time required to perform rejuvenation is obtained. Whether considering the model based or the measurement based analysis, we actually want to get optimal and accurate time to perform rejuvenation. However, the model based analysis ultimately produces a fixed time interval, according to which the software will perform rejuvenation periodically, and the measurement based analysis is based on the system resource consumption, and it is possible to trigger rejuvenation at any time when the running states of system resources are in an emergency. Our studies belong to the measurement based analysis. In both kinds of software aging studies, how to accurately predicting the aging time is definitely an important and indispensable research topic.

In this paper, we propose a novel software aging prediction method using Artificial Bee Colony (ABC) algorithm and BP Neural Network, named as ABC-BP method, which fully considers the influence of the shortcomings of BP neural network on the software aging prediction trend, and improves with the artificial bee colony algorithm. That is, the artificial bee colony algorithm is used to optimize the weight and threshold of the randomly initialized BP neural network. Besides we also compare our method with other representative machine learning algorithms using the Waikato Environment for Knowledge Analysis (WEKA) toolset, and analyze the prediction accuracy of software aging for each algorithm based on different evaluation indicators. Experiment results show that our ABC-BP method fit the prediction trend of software aging more accurately than compared methods, and our method also has faster convergence speed and more stable prediction results.

The rest of paper is organized as follows. Section II presents the preliminaries of artificial bee colony algorithm and BP neural network, together with discussion of related work. Section III describes the details of our ABC-BP method. Section IV presents the experiment process and result analysis. We conclude the paper in the last section.

II. RELATED WORK

A. ARTIFICIAL BEE COLONY ALGORITHM

The artificial bee colony algorithm is an optimization method proposed by imitating bee behavior. In order to solve the multivariate function optimization problem, Karaboga proposed an artificial bee colony algorithm model [6]. The artificial bee colony algorithm has the characteristics of simple operation, less control parameters, high search precision and strong robustness [7], [8].

The smallest search model in which swarm colonies produce group intelligence consists of three basic components, i.e., food sources, employed bees, and unemployed bees. There are two of most basic behavioral models, i.e., recruiting bees for food sources and giving up food source. The value of the food source is determined by the profitability. Employed bees is also known as leader, which corresponds one-to-one with the collected food sources. Leading bees store information about a food source (distance, direction relative to the hive, richness of food source, etc.) and share the information with other bees under a certain probability. Unemployed bees are like to find and mine food sources. There are two types of unemployed bees, i.e., Scouter and Follower. The scouters search for new food sources near the hive [9], and the followers in the hive find the food source by sharing relevant information with the leader. In the process of artificial bee colony algorithm search optimization, the roles of three types of bees are different. That is, the leading bees are used to maintain good solutions, the following bees are used to improve convergence speed, and the scout bees are used to enhance the ability to get rid of the local optimum [10]. The dance area is the most important place to exchange information in the hive. The dance of bees is called swing dance [11]. The position of each food source represents a possible solution to the optimization problem, and the amount of nectar of the food source corresponds to the quality or fitness of the corresponding solution calculated according to the following formula.

$$f(X_i) = \begin{cases} 1 & MSE_i = 0 \\ 1/(MSE_i + 1) & MSE_i > 0 \end{cases} \quad (1)$$

where: $i = 1, \ldots, Ns$; MSE_i is the BP network mean square error of the ith solution.

In this algorithm, the number of leading bees (Ne) equals to the number of followers (No), and also equals to the number of solutions (Ns) in the population. First, the artificial bee colony algorithm generates sn initial solutions (sn is the number of food sources). Each solution X_i ($i = 1, 2, \ldots, sn$) is a vector of d dimensions, and d is the number of optimization parameters. Then, the leader, the follower, and the scouter begin a looping search with the number of loops being MCN. Leaders adopt the greedy criterion, compare the optimal solution and the neighborhood search solution in memory, and replace the memory solution when the search solution is better than the memory optimal solution; otherwise, it remains unchanged. After all the lead bees complete the neighborhood search, the bee jumps to the tail dance and shares the honey source information with the follower. The followers select the honey source according to the honey source information with a certain probability. The bees producing more honey will attract more followers with a larger probability. Similarly, the followers search in the neighborhood of the honey source using the greedy criterion, and compare the followers search solution with the original bee solution. If the search solution is better than the original bee solution, they will replace the original bee solution and change the role of each other; otherwise, it remains unchanged. Follower select the food source according to the probability value P_i, P_i is calculated according to the following formula.

$$P_i = f(X_i) / \sum_{n=1}^{Ns} f(X_n) \quad (2)$$

Leaders and followers perform neighborhood search according to the following formula.

$$V_{ij} = X_{ij} + rand(-1, 1)(X_{ij} - X_{kj}) \quad (3)$$

where: i is the number of the solution, $k \in \{1, 2, \ldots, Ns\}$ is randomly generated, and $i \neq k$. $J \in \{1, 2, \ldots, D\}$.

For the scenario that leading bees fall into the local optimum, it should keep the number of iteration, i.e., $Limit$ as important control parameter in the artificial bee colony algorithm, not be changed. When the fitness of the leader is not the current global optimum, the food source is abandoned and replaced with the food source randomly searched by the scouters. Let the abandoned solution be the Xi, the scouters use the following formula to generate a new solution instead.

$$X_i = X_{min} + rand(0, 1)(X_{max} - X_{min}) \quad (4)$$

Though there have been some optimizations for artificial bee colony algorithms, such as the impact of parameter settings on the performance [12], in this paper, we use the basic artificial bee colony algorithm to construct our ABC-BP method, which could get satisfactory results.

B. BP NEURAL NETWORK

BP (Back Propagation) neural network, the learning process of error back propagation algorithm, consists of two processes, i.e., forward propagation of information and back propagation of errors. The BP neural network has three layers, namely input layer, hidden layer, and output layer. The input signal is processed layer by layer from the input layer through the hidden layer to the output layer. The neuron state of each layer only affects the state of the next layer of neurons. If the output layer does not get the expected output, it goes to back propagation, and adjusts the network weight and threshold according to the prediction error, so that the BP neural network prediction output is continuously approaching the expected output. BP neural network must train the network before the prediction, and the network has the associative memory and prediction ability through training. The BP neural network training process includes the following steps.

1) **Neural network initialization.** According to the system input and output sequence (X, Y), the number of network input layer nodes n, the number of hidden layer nodes l, the number of output layer nodes m, and the connection weight between the input layer, the hidden layer and the output layer neurons are determined. Initialize the hidden layer threshold a, the output layer threshold b, the given *learning rate*, and the *neuron excitation function*.
2) **Hidden layer output calculation.** The hidden layer output H is calculated from the input vectors X, w and a.
3) **Output layer output calculation.** The BP neural network predictive output O is calculated according to the hidden layer output H, the connection weight w and the threshold b.
4) **Error calculation.**The network prediction error e is calculated based on the network prediction output O and the expected output Y.
5) **Weight update.**The network connection weight is updated according to the network prediction error e.
6) It is judged whether the algorithm iteration ends or not. If not, return to *step 2*.

C. SOFTWARE AGING PREDICTION

In previous studies, machine learning algorithms have been widely used in the prediction of software aging. In [13], the M5P algorithm was applied to a variety of complex software aging scenarios. It achieved the required error accuracy range, and thus provided a way to prevent software aging for Java EE. In [14], the decision tree algorithm, LDA/QDA (linear and quadratic discriminate analyses) algorithm, naive Bayes algorithm, support vector machine algorithm, k-nearest neighbor algorithm and random forest were compared by different software aging prediction scenarios. The prediction accuracy of the algorithm shows that the random forest algorithm has the best prediction results. On this basis, the author further simplified the collected data by using the regularization technique of the cable, and the results showed that the ideal prediction effect can still be achieved. Sudhakar *et al.* [15] predicted the software aging phenomenon in the cloud system, considering the number of processes, the number of threads, the CPU utilization, the number of open TCP connections, the available physical memory, and the available exchange areas. The feed-forward neural network structure was used as a training model, and training was performed until the output error is minimized and the training stopped. In [16], seven machine learning algorithms, i.e., linear regression, minimum median square, Gaussian process, M5P, REP tree, sequence minimum optimization algorithm and multilayer perception neural network were selected and implemented in four aging scenarios. The predictions show that the multilayer perception neural network has the highest accuracy. In these typical machine learning algorithms, the multilayer perception neural network standed out and showed its superiority in prediction error. Therefore, we also decided to use BP neural network as the main body of the training model, and then optimize it. In [17], the three performance indicators of CPU utilization, memory utilization and available exchange area were selected, and given different weights according to the influence of each indicator, analyzed and predicted by multiple linear regression algorithm. In [18], the time series and artificial neural network were combined to predict the three software aging factors of response time, memory usage and swap space. An evaluation was performed to illustrate the effectiveness of this method. In [19], the classification algorithms in machine learning, such as Bayesian, decision tree and support vector machine, were mainly used to model the aging process. These machine learning algorithms show good predictive effects when considering cost.

In [20], the software aging of real video on demand system was analyzed. The principal component analysis method was used to reduce the input variables, and the artificial neural network was used for prediction. In [21], by constructing a multilayer back-propagation neural network and taking the workload as input, the results showed that the prediction effect is good and the workload is proved to impact the software of aging. In [22], a gray correlation artificial neural network (GRANN_ARIMA) method was proposed to mix linear and nonlinear models to predict software aging. In [23], the decision tree algorithm, support vector machine algorithm (SVM) and deep belief network algorithm (DBN) are used to predict the software aging of the Android operating system. The experimental results show that the deep belief network algorithm was an effective prediction method when the data volume reaches a certain level. In [24], the memory aging indicator was used as the only indicator of prediction, and a classification algorithm was used to predict a

commercial tool called Plumbr, thereby improving the quality of memory leak detection.

D. ABC BASED ANN

In fact, the combination of artificial bee colony algorithm and neural network has emerged in some areas. In [25], an improved neural network algorithm based on artificial bee colony algorithm and its application in wastewater treatment was proposed. By combining the two algorithms, the accuracy of water quality evaluation was successfully improved. In [26], artificial bee colony algorithm combined with hybrid neural network was proposed to predict natural gas consumption. The experimental results showed that the error rate was low. In [27], an improved artificial bee colony algorithm combined with neural network was proposed for short-term wind speed prediction. Through case analysis, we can see that the results were more accurate than traditional neural network prediction. From the above analysis, we find that the combination of artificial bee colony algorithm and neural network has not applied into the software aging prediction filed for IoT environments.

III. ABC-BP METHOD

With the increasing number of IoT devices, the amount of various data are also increasing sharply, and the software aging problem in IoT platforms is becoming more and more obvious and serious. The central idea of ABC-BP method is to optimize the BP neural network during software aging prediction, so as to improve the accuracy of prediction. The specific operation process of our method are presented in this section.

A. CORE IDEA

In the ABC-BP, our main purpose is to determine when a fault has occurred by monitoring the system aging indicators. The workflow of the ABC-BP is shown in Figure 1. First, we input the data that we want to monitor into the BP neural network. Here we determine the correlation coefficient of the data according to the time series analysis to determine the number of nodes in the input layer. At this point we built an initial BP neural network. BP neural network automatically initializes a set of weights and thresholds after running. We enter this weight and threshold as initial values into the artificial bee colony algorithm, which will select a set of best solutions for us based on fitness. we use this solution to find the optimal weights and thresholds for BP neural networks, then continue to train the BP neural network. After the training is completed, the data are simulated and predicted. Through several simulations, results of less error with the actual data are obtained, so as to achieve the effect of predicting software aging.

The ABC-BP method combines the characteristics of the global iteration of the artificial bee colony algorithm and the generalization mapping ability of the neural network. It solves the shortcomings of BP neural network which is easy to fall into the local optimal solution and the slow convergence speed. It better fits the software aging prediction trend, and proves that this method has better stability through multiple simulations.

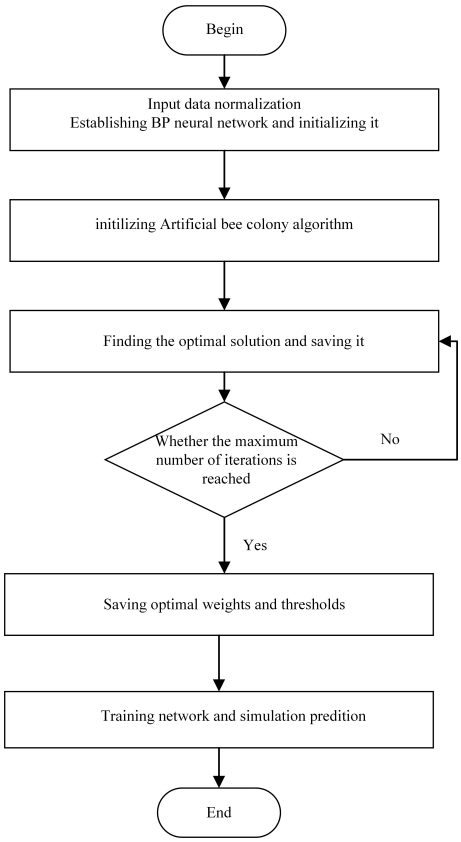

FIGURE 1. The workflow of ABC-BP method.

B. MODEL CONSTRUCTION

The ABC-BP model used to predict software aging is mainly divided into three parts, Firstly, a BP neural network needs to be constructed as the main body. Secondly, artificial bee colony algorithm is added to optimize the weight and threshold of BP neural network. Finally, the experimental results and real values of ABC-BP method are used.

1) **BP neural network initialization.** Normalize the input data to prevent the network from converge due to singular samples and speed up the convergence. A BP neural network is established and initialized, and a set of weights and thresholds are randomly initialized in the neural network, the initial weights and thresholds are entered into the artificial bee colony algorithm.
2) **Artificial bee colony algorithm initialization and optimization process.** Setting the number of solutions (*Ns*), limit value (*Limit*), maximum number of cycles (*MCN*), the above weights and thresholds are used as the initial solution of the artificial bee colony algorithm, and use the artificial bee colony algorithm to search for the optimal solution in the neighborhood using formula (3). The fitness value is calculated using equation (2) and the optimal solution is saved. If you

find a better solution, give up this solution and save a better solution. If it is found that the optimal solution is not updated within the limit value, indicating that the local optimal solution is trapped at this time, the new solution is searched again according to formula (4). At this point, it is judged whether the optimization process reaches the maximum number of iterations. If it is not reached, the search for the optimal solution is continued, otherwise the optimal solution is saved.

3) **Simulation prediction.** The BP neural network is trained by finding an optimal set of weights and thresholds through the optimal solution. After the training, the simulation test is performed, and the output is the prediction result of the software aging index.

C. DISCUSSION OF OUR ABC-BP MODEL

ABC algorithm is a new type of intelligent random search method. With its powerful global search ability, it can solve the local mechanism problem of BP neural network. Therefore, the ABC algorithm is used to optimize the weight and threshold of the neural network, and the global search ability of the ABC algorithm can be effectively combined with the local search ability of the BP neural network, which can effectively improve the convergence speed and training accuracy of the network. Thereby improving the security of the IoT system and effectively controlling the software aging problem of its background server. Not only that, after several experiments, we found that our ABC-BP prediction method is more stable, and the results of multiple experiments are not biased, and BP neural network needs to be improved at this point. Compared with other machine learning algorithms, the prediction accuracy of the ABC-BP neural network is also quite high enough.

IV. EXPERIMENT AND RESULTS ANALYSIS

It should be explained that the IoT environment mainly consists of three parts, i.e., cloud data center with many computing servers, mobile terminals or sensors to collect data, and communication networks. Lots of IoT business systems are currently built on top of the cloud platform for efficient computing. Among all the software systems running in servers or terminals, our studies pay more attentions to software systems running in the IoT data center for data computing and analysis. Therefore, the experimental dataset we use in this paper are from cloud server that could deal with IoT applications. We compare our method with the traditional BP neural network in MATLAB environment, and analyze the output results. Then we compare our method with current popular machine learning algorithms and observe the aging evaluation indicators to show our contributions.

A. SELECTION OF DATA SAMPLES

The data set used in this experiment was provided by Google [28]. This dataset contains a total of 1600 subfiles, which are monitored using different virtual machines. Each virtual machine contains a data set of ten days, where each piece of data are recorded by the virtual machine monitor every 5 minutes, so 288 pieces of data per day. Each piece of data contain two columns, i.e., CPU utilization and memory utilization. Since the main reason for software aging is the consumption of system resources, memory usage is one of the important indicators. Therefore, our experiments only select the second column of dataset, that is, memory utilization as the aging indicator of this experiment.

In order to simulate the process of increasing memory consumption, we selecte data of seven days and integrate the seven sub-data sets to form a data set whose memory is gradually exhausted. This dataset contains a total of 2016 pieces of data, where the data from the first six days are used as the training set and rest data from last day are used as the test set.

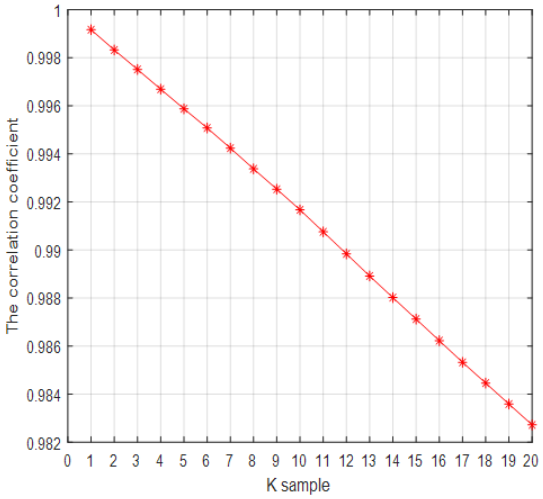

FIGURE 2. Correlation coefficients of the first 20 samples.

B. EXPERIMENT DESIGN

In this experiment, the parameters of the bee colony algorithm are set to NS=100, $Limit$=200, MCN=100. The input layer of the BP neural network has 3 nodes, and the output layer has 1 node. The number of input layer nodes is determined by the correlation coefficient of the first 20 samples in the training set, as shown in Figure 2. It can be seen that the correlation coefficient of the data is getting smaller and smaller. We want to use the data with better correlation as input, so 3 nodes are used in the input layer. The hidden layer has 2 layers, each layer has 5 nodes, the tansig function is the transfer function from the input layer to the first hidden layer and the first hidden layer to the second hidden layer neuron, and the purelin function is used as the second hidden layer to output layer transfer function. The traingd function is used as the training function. The network has a maximum number of trainings of 2000, an expected error of $1 \times e^{-4}$, and a learning rate of 0.25.

C. EXPERIMENT RESULTS AND ANALYSIS

We analyze the fitness of the artificial bee colony algorithm to determine the degree of optimization of the weight

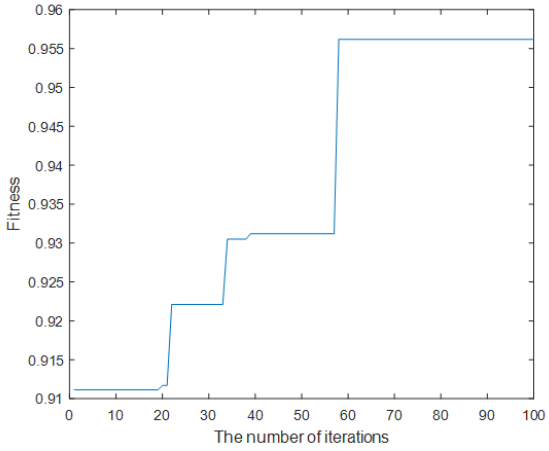

FIGURE 3. The fitness of the bee colony solution.

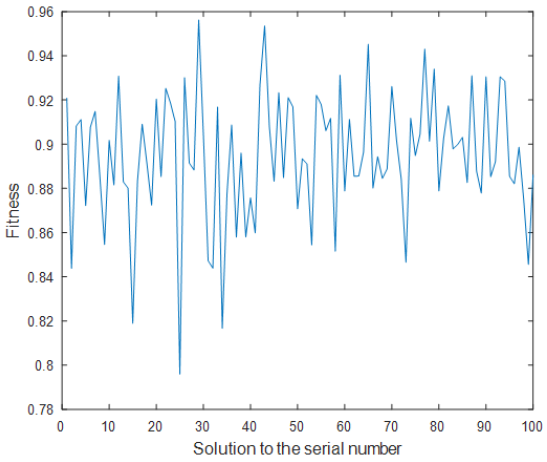

FIGURE 4. The fitness of all final solutions of the bee colony.

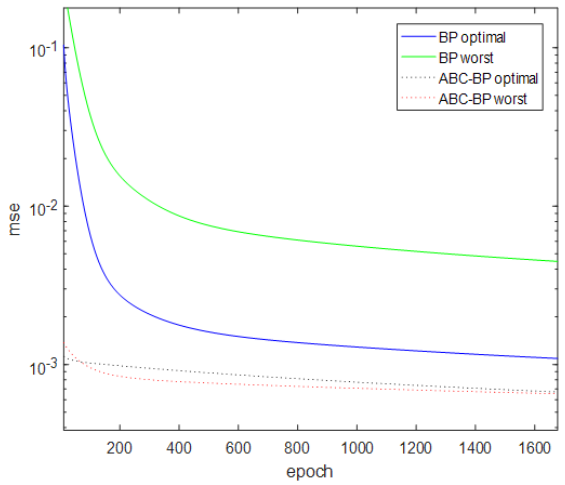

FIGURE 5. Error comparison.

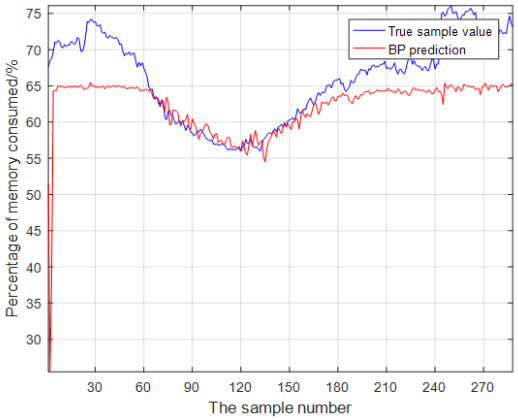

FIGURE 6. Comparison of BP predicted values with real values.

and threshold. Using the mean square error (MSE), we make comparison of the real value and the predicted value to BP neural network, and ABC-BP method is compared to illustrate the prediction effect on software aging. The fitness of the bee colony solution and the fitness of all final solution of the bee colony are shown in Figures 3 and 4.

As can be seen from Figure 3, we selected the fitness results of a bee colony solution, a total of 100 iterations. In the first 60 iterations, we can see that the fitness value is increasing and the optimization results are getting better. After 60 times, the result tends to be stable. At this time, we see that the fitness value is above 0.955. This is a very good fitness value, so we can judge that the result of the solution has been optimized. Figure 4 is the fitness value of all solutions. From the figure, we can see the fitness results of a total of 100 solutions. Due to the richness of food sources and other factors, the degree of the obtained solutions is not the same. Nevertheless, the minimum fitness value of this experiment is above 0.8, and the optimal value is basically 0.98. Since we use the optimal fitness value after comparison as the result of the final solution of this experiment, we can expect the result to be quite satisfactory.

The test set is the input into the trained model, and the obtained error comparison with the comparison between the true value and the test value are as shown in the Figure 5. It can be seen that when the mean square error is 10^{-3}, compared with the BP neural network, the ABC-BP neural network can meet the requirements after less than 100 iterations, and the BP neural network passes through. After 200 iterations, it stabilized, but the results were not ideal enough to meet our requirements. Therefore, our method not only has a faster convergence speed, but also has a smaller error value than the BP neural network. It can be seen that ABC-BP neural network is a more effective and accurate method than BP neural network in predicting software aging.

Figure 6 and Figure 7 show the prediction of software aging using BP neural network and ABC-BP neural network. The software aging index and data amount are the same. It can be seen from Figure 6 that the actual value of memory utilization in the initial stage is about 68%, and the predicted result is about 65%. The predicted value in Figure 7 is about 67%, which is basically consistent with the actual value. As the prediction continues to deepen, the coincidence distance between the true value and the predicted value in Figure 6

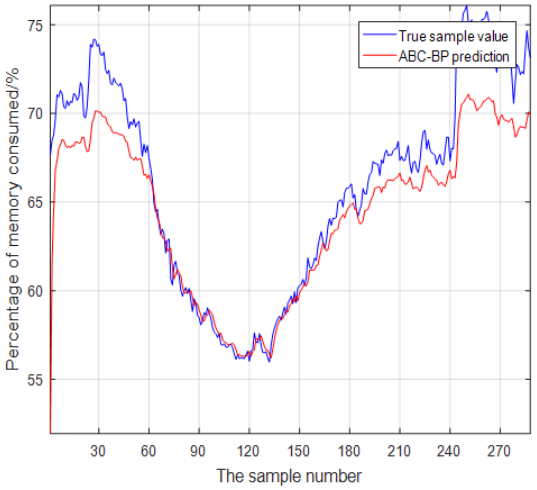

FIGURE 7. Comparison of ABC-BP predicted values with real values.

TABLE 1. Comparison with representative machine learning algorithms.

	Correlation coefficient	MAE	RMSE	RAE
LR	0.9706	3.7236	4.5681	24.3963%
BP	0.9688	4.4338	5.2557	29.0493%
GP	0.9706	14.049	16.3918	92.0471%
ABC-BP	0.8641	2.6896	2.4697	22.9147%
SMO	0.9706	3.714	4.5696	24.3336%

is significantly less than that in Figure 7. It can be seen that the experimental results are closer to the true value. After 210 sample values, we can see that the prediction results of BP neural network are basically maintained around a fixed value, while the ABC-BP neural network has consistent with the trend of the true value curve. The predicted value is closer to the true value, which is the better result of the software aging prediction. Not only that, after many runs, we found that the ABC-BP method results are more stable. The reason is because the weights of the BP neural network are randomly initialized. The BP neural network is extremely sensitive to the initial value selection, so it has a great impact on the results. This also shows the superiority of the ABC-BP neural network method for software aging prediction.

In addition, we also use WEKA to compare the current popular algorithms with the ABC-BP method. As an open data mining work platform, it integrates a large number of machine learning algorithms that can undertake data mining tasks, including preprocessing, classification, regression, clustering, association rules and visualization on new interactive interfaces. In our experiments, 85 percent of the collected data was used for training and 15 percent was used for testing. The data attributes are still the memory utilization. In order to predict the software aging time, we add a time attribute later. The evaluation indicators are Correlation coefficient, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Relative Absolute Error (RAE). The experimental results are shown in the Table 1.

It can be seen from Table 1 that the linear regression (LR), Gaussian process (GP) and Sequential Minimal Optimization (SMO) machine learning algorithm have the best correlation coefficients with value 0.9706. In the column of average absolute error, it is very unsatisfactory to see the Gaussian process, and the difference between the results of other methods is obvious, which indicates that the algorithm is not good for the processing of this data. Our method has the best results in all algorithms and the most accurate prediction. The latter two evaluation indicators are basically the same as the average absolute error, and the ABC-BP shows the best experimental results. Therefore, it can be seen that in addition to the correlation coefficient, other evaluation indicators have a certain improvement compared with other machine learning algorithms, indicating the superiority of our ABC-BP method.

V. CONCLUSION

Predicting the software aging accurately is quite important and indispensable for improving the software running reliability of IoT applications. In this paper, we propose the ABC-BP software aging prediction method, where the artificial bee colony algorithm is used to optimize the weight and threshold of the randomly initialized BP neural network. Experiment results show that our method fit the prediction trend of software aging more accurately than traditional BP neural network, and our method also has faster convergence speed and more stable prediction results.

REFERENCES

[1] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, "Software rejuvenation: Analysis, module and applications," in *Int. Symp. Fault-Tolerant Comput. Dig. Papers*, Jun. 1995, pp. 381–390. doi: 10.1109/FTCS.1995.466961.

[2] H. Okamura, J. Zheng, and T. Dohi, "A statistical framework on software aging modeling with continuous-time hidden Markov model," in *Proc. IEEE 36th Symp. Reliable Distrib. Syst.*, Oct. 2017, pp. 114–123. doi: 10.1109/SRDS.2017.24.

[3] H. Meng, X. Hei, Y. Li, Y. Du, and G. Xie, "A rejuvenation model for software system under normal attack," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2015, pp. 1160–1164. doi: 10.1109/Trustcom-BigDataSe-ISPA.2015.498.

[4] O. Nhway, "Reliability modeling and analysis of application servers using stochastic Petri Net Model," in *Proc. 7th Int. Conf. Adv. Inf. Manage. Service (ICIPM)*, Nov. 2011, pp. 163–167.

[5] F. Salfner and K. Wolter, "A Petri net model for service availability in redundant computing systems," in *Proc. IEEE Winter Simul. Conf. (WSC)*, Dec. 2009, pp. 819–826. doi: 10.1109/WSC.2009.5429681.

[6] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Dept. Comput. Eng., Eng. Faculty, Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, Jun. 2005, pp. 131–142.

[7] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, 2009. doi: 10.1016/j.amc.2009.03.090.

[8] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, Aug. 2008. doi: 10.1016/j.asoc.2007.05.007.

[9] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Nov. 2007. doi: 10.1007/s10898-007-9149-x.

[10] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Inf. Sci.*, vol. 192, no. 1, pp. 120–142, Apr. 2012. doi: 10.1016/j.ins.2010.07.015.

[11] V. Tereshko and A. Loengarov, "Collective decision-making in honey bee foraging dynamics," *Comput. Inf. Syst. J.*, vol. 9, pp. 324–356, Sep. 2005.

[12] B. Akay and D. Karaboga, "Parameter tuning for the artificial bee colony algorithm," in *Proc. Int. Conf. Comput. Collective Intell., Semantic Web, Social Netw. Multiagent Syst.* Berlin, Germany: Springer-Verlag, Oct. 2009, pp. 608–619. doi: 10.1007/978-3-642-04441-0_53.
[13] J. Alonso, J. Torres, J. L. Berral, and R. Gavaldà, "Adaptive on-line software aging prediction based on machine learning," in *Proc. IEEE Int. Conf. Dependable Syst. Netw.*, May 2010, pp. 507–516. doi: 10.1109/DSN.2010.5544275.
[14] J. Alonso, L. Belanche, and D. R. Avresky, "Predicting software anomalies using machine learning techniques," in *Proc. IEEE 10th Int. Symp. Netw. Comput. Appl.*, Oct. 2011, pp. 163–170. doi: 10.1109/NCA.2011.29.
[15] C. Sudhakar, I. Shah, and T. Ramesh, "Software rejuvenation in cloud systems using neural networks," in *Proc. IEEE Int. Conf. Parallel, Distrib. Grid Comput.*, Dec. 2015, pp. 1356–1357. doi: 10.1109/PDGC.2014.7030747.
[16] M. Yakhchi, J. Alonso, M. Fazeli, A. A. Bitaraf, and A. Patooqhy, "Neural network based approach for time to crash prediction to cope with software aging," in *Proc. J. Syst. Eng. Electron.*, vol. 26, no. 2, pp. 407–414, Feb. 2015. doi: 10.1109/JSEE.2015.00047.
[17] S. Jia, C. Hou, and J. Wang, "Software aging analysis and prediction in a Web server based on multiple linear regression algorithm," in *Proc. IEEE 9th Int. Conf. Commun. Softw. Netw.*, May 2017, pp. 1452–1456. doi: 10.1109/ICCSN.2017.8230349.
[18] H. El-Shishiny, S. S. Deraz, and O. B. Badreddin, "Mining software aging: A neural network approach," in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2008, pp. 182–187. doi: 10.1109/ISCC.2008.4625660.
[19] A. Andrzejak and L. Silva, "Using machine learning for non-intrusive modeling and prediction of software aging," in *Netw. Oper. Manage. Symp. (NOMS)*, May 2008, pp. 25–32. doi: 10.1109/NOMS.2008.4575113.
[20] X. Du, C. Xu, D. Hou, and Y. Qi, "Software aging estimation and prediction of a real VOD system based on PCA and neural networks," in *Proc. IEEE Int. Conf. Inf. Automat.*, Jun. 2009, pp. 111–116. doi: 10.1109/ICINFA.2009.5204903.
[21] K.-X. Xue, L. Su, Y.-F. Jia, and K.-Y. Cai, "A neural network approach to forecasting computing-resource exhaustion with workload," in *Proc. 9th Int. Conf. Qual. Softw.*, Aug. 2009, pp. 315–324. doi: 10.1109/QSIC.2009.48.
[22] R. Sallehuddin, S. M. Shamsuddin, and S. Z. M. Hashim, "Hybridization model of linear and nonlinear time series data for forecasting," in *Proc. 2nd Asia Int. Conf. Model. Simul. (AMS)*, May 2008, pp. 597–602. doi: 10.1109/AMS.2008.142.
[23] S. Huo, D. Zhao, X. Liu, J. Xiang, Y. Zhong, and H. Yu, "Using machine learning for software aging detection in Android system," in *Proc. 10th Int. Conf. Adv. Comput. Intell.*, Jan. 2018, pp. 741–746.
[24] V. Sor, P. Oü, T. Treier, and S. N. Srirama, "Improving statistical approach for memory leak detection using machine learning," in *Proc. IEEE Int. Conf. Softw. Maintenance*, Sep. 2013, pp. 544–547. doi: 10.1109/ICSM.2013.92.
[25] D. Pan and J. Cao, "An improved neural network algorithm based on artificial bee colony algorithm and its application in sewage treatment," in *Proc. Int. Conf. Behav., Econ. Socio-Cultural Comput. (BESC)*, Oct./Nov. 2015, pp. 83–88. doi: 10.1109/BESC.2015.7365963.
[26] M. Akpinar, M. F. Adak, and N. Yumusak, "Forecasting natural gas consumption with hybrid neural networks—Artificial bee colony," in *Proc. IEEE Int. Conf. Intell. Energy Power Syst.*, Jul. 2016, pp. 35–58. doi: 10.1109/IEPS.2016.7521852.
[27] G. Jia, D. Li, L. Yao, and P. Zhao, "An improved artificial bee colony-BP neural network algorithm in the short-term wind speed prediction," in *Proc. 12th World Congr. Intell. Control Automat. (WCICA)*, Jun. 2016, pp. 2252–2255. doi: 10.1109/WCICA.2016.7578265.
[28] C. Reiss *et al.*, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. ACM Symp. Cloud Comput.*, Oct. 2012, pp. 1–13. doi: 10.1145/2391229.2391236.

JING LIU (M'14) received the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2011. He was a Visiting Scholar with the University of Melbourne, from 2014 to 2015. He is currently an Associate Professor of computer science and technology with Inner Mongolia University. He has published more than 20 papers in international conferences and journals. His research interests include software fault tolerance, cloud computing, and the IoT applications.

LINGZE MENG received the B.S. degree in software engineering from the Inner Mongolia University of Science and Technology, in 2017. He is currently pursuing the M.S. degree with the College of Computer Science, Inner Mongolia University. His research interests include software aging and rejuvenation.