Software Reliability Prediction Using Artificial Techniques

Rita G. Al gargoor [1], Nada N. Saleem [2]

**[1] Software Engineering Department, Mosul University, College of Computer sc. And Mathematics
Mosul, ZIP/+964, Iraq**

**[2] Software Engineering Department, Mosul University, College of Computer sc. And Mathematics
Mosul, ZIP/+964, Iraq**

Abstract

Due to the growth in demand for software with high reliability and safety, software reliability prediction becomes more and more essential. Software reliability is a key part of software quality. Over the years, many software reliability models have been successfully utilized in practical software reliability engineering, however, no single model can obtain accurate prediction for all cases. So in order to improve the accuracy of software reliability prediction the proposed model combine the software reliability models with the neural networks (NN). Particle swarm optimization (PSO) algorithm has been chosen and applied for learning process to select the best architecture of the neural network. The applicability of the proposed model is demonstrated through three software failure data sets. The results show that the proposed model has good prediction capability and more applicable for software reliability prediction.

Keywords: *software reliability prediction, neural network, particle swarm optimization*

1. Introduction

As the use of software is increasing, the failures are also increasing rapidly. The consequences of failures may lead to loss of life or economic loss. So, the software professionals need to develop software systems which are not only functionally attractive but also safe and reliable [6]. Software reliability is defined as the probability of failure free software operation for a specified period of time in a specified environment[5]. Being able to predict the number of faults resides in software helps significantly in specifying/ computing the software release day and manage project resources (i.e. people and money) [23].

Software reliability growth models, refers to those models that try to predict software reliability from test data. These models try to show a relationship between fault detection data (i.e. test data) and known mathematical functions such as logarithmic or exponential functions. The goodness of fit of these models depends on the degree of correlation between the test data and the mathematical function[4]. Typically two broad categories of software reliability growth models (SRGMs) include parametric models and nonparametric models. Most of the parametric models are based on nonhomogeneous Poisson process (NHPP) that has been widely used successfully in practical software reliability engineering[24]. Artificial neural network (ANN) with software reliability models have aroused more research interest .

in this paper, we use the effect of neural network to build non-parametric model for software reliability prediction, with the particle swarm optimization (PSO) algorithm used in our work for learning and to select the best architecture of the neural network.

The rest of this paper is organized as follows:
In section 2 a brief review of the researches carried out in the area of software reliability prediction is presented. Section 3 include background about software reliability. In section 4 the various artificial techniques that are applied in this paper are described briefly. Section 5 depicts neural network based approach for software reliability modeling. Section 6 presents the proposed model. Section 7 presents the experimental methodology and results. Section 8 conclude the paper.

2. Related Work

In recent years, many papers have presented various models for software reliability prediction . In this section, some works related to neural network techniques for software reliability modeling and prediction are presented.

Karunanithi et al. [18][16] first presented neural network based software reliability model to predict cumulative number of failures, the execution time is used as the input of the neural network. They used different networks like Feed Forward neural networks, recurrent neural networks like Jordan neural network and Elman neural network in their approach.
Karunanithi et al. [19] also used connectionist models for software reliability prediction, the results shows that the connectionist models may adapt well across different datasets and exhibit a better predictive accuracy.
Karunanithi et al. [17] they also predict the software reliability using neural network and present a solution to the scaling problem uses a clipped linear unit in the output layer of the neural network.

Su et al.[26] they used the neural network approach to build a dynamic weighted combinational model.
Sitte [25] presented a neural network based method for software reliability prediction. He compared the approach with recalibration for parametric models using some meaningful predictive measures with same datasets. They concluded that neural network approach is better predictors.
Cai et al. [8] proposed a neural network based method for software reliability prediction. They used back propagation algorithm for training. They used multiple recent 50 failure times as input to predict the next-failure time as output. They evaluated the performance of the approach by varying the number of input nodes and hidden nodes. They concluded that the effectiveness of the approach generally depends upon the nature of the handled data sets.
Hu et al. [12] proposed an artificial neural network model to improve the early reliability prediction for current projects/releases by reusing the failure data from past projects/releases.
Su et al. [27] proposed a dynamic weighted combinational model (DWCM) based on neural network for software reliability prediction. They used different activation functions in the hidden layer depending upon the software reliability growth models (SRGM).
Aljahdali et al. [2] investigated the performance of four different paradigms for software reliability prediction. They presented four paradigms like multi-layer perceptron neural network, radial-basis functions, Elman recurrent neural networks and a neuro-fuzzy model. They concluded that the adopted model has good predictive capability.

In [24], Singh et al. used feed forward neural network for software reliability prediction. They applied back propagation algorithm as learning algorithm. The experimental result had shown that the proposed system has better prediction than some traditional software reliability growth models.
Wang et al.[30] they used artificial neural network with software reliability combinational model by constructing the transfer function corresponding to the selected model.

In [13], Huang et al. derived software reliability growth models (SRGM) based on non-homogeneous poison processes (NHPP) using a unified theory by incorporating the concept of multiple change-points into software reliability modeling. They estimated the parameters of their proposed models using three software failure data sets and compared results with some existing SRGM. Their model predicted the cumulative number of failures in various stages of software development and operation.

In [6][15][31], it was presented that the performance of a neural network system can be significantly improved by combining a number of neural networks.

3. Background

3.1 Concept of software reliability

Software reliability is one of the important factor been considered while ensuring the software quality. In simple term we can say that software reliability deals with the failure or faults that exist in the system [11]. Failures are the result of a fault in the software code, and several failures can be the result of one fault. The process of finding and removing faults to improve the software reliability can be described by a mathematical relationship called a software reliability growth model (SRGM)[3].

3.2 Software reliability growth models(SRGMs) and criteria

A software reliability growth model (abbreviated as SRGM) is known as one of the fundamental technologies for quantitative software reliability assessment, and playing an important role in software project management for producing a highly-reliable software system[14]. SRGM is mathematical model, shows how software reliability improves as faults are detected and repaired. SRGM can be used to predict when a particular level of reliability is likely to be attained. Thus, SRGM is used to determine when to stop testing to attain a given reliability level [21]. There are many software reliability growth models but the Commonly used model of software reliability models are JM, GO model, MO model, Sch model, S-Shape model. To evaluate the prediction powers of different models, it is necessary to use a meaningful measures. In this paper we use two criteria: Root Mean Square Error (RMSE) and Average Error(AE). These criteria are used to measure the difference between the actual and predicted values, the formulas is as Eq. (1) (2).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\text{c(k)} - \hat{c}\text{(k)}\right)^2} \qquad (1)$$

$$\text{AE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{c}(k) - c(k)}{c(k)}\right| \times 100 \qquad (2)$$

Where n is the number of groups of failure data, $c(k)$ is the number of the actual failures in each group of failure data, $\hat{c}(k)$ is the number of the predicted failures. The smaller the RMSE and AE, the stronger that the model prediction ability[6] [30].

4. Overview of The Artificial Techniques Used

4.1 Artificial neural networks (ANNs)

An artificial neural network, or simply neural network, is a type of artificial intelligence (computer system) that

attempts to mimic the way the human brain processes and stores information. It works by creating connections between mathematical processing elements, called neurons, Fig.1 shows a neuron. Knowledge is encoded into the network through the strength of the connections between different neurons, called weights, and by creating groups, or layers, of neurons that work in parallel. The system learns through a process of determining the number of neurons or nodes and adjusting the weights for the connections based upon training data[27][29].

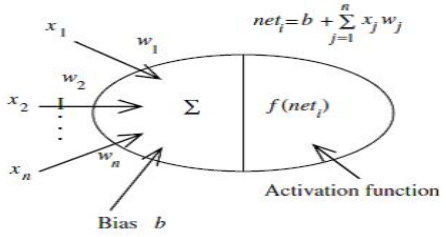

Fig. 1 The model of neuron

4.2 Particle swarm optimization(PSO)

Particle swarm optimization (PSO) is a population-based stochastic optimization technique modeled on the social behaviors observed in animals or insects, e.g., bird flocking, fish schooling, and animal herding. It was originally proposed by James Kennedy and Russell Eberhart in 1995. Since its inception, PSO has gained increasing popularity among researchers and practitioners as a robust and efficient technique for solving difficult optimization problems[7].

The swarm of particles initialized with a population of random candidate solutions move through the d-dimension problem space to search the new solutions. The fitness f, can be calculated as the certain qualities measure. Each particle has a position represented by a position-vector present i (i is the index of the particle), and a velocity represented by a velocity-vector velocity i. After every iteration the best position-vector among the swarm is stored in a vector. The update of the velocity from the previous velocity to the new velocity is determined by Eq. (3). The new position is then determined by the sum of the previous position and the new velocity by Eq. (4).

Velocity ij(new) = w * velocity ij(old) + c1rand1(pbest ij(old)) − present ij(old) + c2rand2(gbest j(old) − present ij(old)) (3)

Present ij(new) = present ij(old) + velocity ij(new) (4)

Here w is the inertia weight, rand1 and rand2 are the random numbers usually chosen between [0,1]. c1 is a positive constant ,called as coefficient of the self-recognition component, c2 is a positive constant, called as coefficient of the social component and the choice of value is c1=c2= 2 generally referred to as learning factors[28].

4.3 Training the artificial neural network(ANN) using PSO algorithm

In this paper we use the particle swarm optimization(PSO) algorithm for learning and selecting the best architecture of our feedforward neural network, PSO is applied to feedforward neural network as follows:

The position of each particle in swarm represents a set of weights for the current epoch or iteration. The dimensionality of each particle is the number of weights associated with the network. The particle moves within the weight space attempting to minimize learning error. Changing the position means updating the weight of the network in order to reduce the error of the current epoch. In each epoch, the particles update their position by calculating the new velocity, and move to the new position. The new position is a set of new weights used to obtain the new error. This process is repeated and the particle with the lowest learning error is considered as the global best particle. The training process continues until satisfactory error is achieved by the best particle or computational limits (maximum iteration) are exceeded[1].

5. Neural Network Based Approaches for Software Reliability Modeling

5.1 The selection of the base models

As we mentioned in section 3 there are many software reliability models, among these models we choose GO model, logistic curve model, S-Shape model because their performance in software reliability evaluation. The failure mean value function of These three model are as Eq. (5)(6) and (7) [22].

$$m(t) = a(1 - e^{-bt}) \quad \text{GO model} \quad (5)$$

$$m(t) = \frac{a}{1 + ke^{-bt}} \quad \text{logistic curve model} \quad (6)$$

$$m(t) = a(1 - (1 + bt)e^{-bt}) \quad \text{S-Shape model} \quad (7)$$

5.2 Design the activation function of a neural Network for each base model

The derivation of the neural network into a software reliability modeling had depicted as follow:

For example, if we take the logistic growth curve model, this model simply fits the mean value with a form of the logistic function. so if we consider the basic feedforward neural network shown in Fig. 2, the derivation equations for the activation function of the neural network into software reliability models are depicted in Eq. (8),(9) and (10) .

$$y(t) = w_{11}^0 \left(1 - e^{-(w_{11}^1 t)}\right) \quad \text{GO model} \quad (8)$$

$$y(t) = \frac{w_{11}^0}{1+e^{-(w_{11}^1 t)}} \quad \text{logistic curve model} \quad (9)$$

$$y(t) = w_{11}^0 \left(1 - (1 + w_{11}^1 t)\right) e^{-w_{11}^1 t} \quad \text{S-Shape model} \quad (10)$$

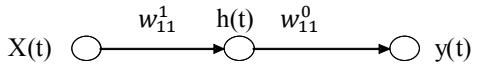

Fig. 2 Feed-forward neural network with single neuron in each layer

Where the input, x(t), at time t is fed to the input layer, h(t) is the output of the hidden layer, y(t) is the output of the output layer[26].

6. Methods

In this section, we present the proposed model for software reliability prediction based on neural network.

6.1 Software reliability data

During the execution process the software may fail. Fig. 3 illustrate the software failure process, where t_i is the execution time for i th software failure and $\Delta t_i = t_i - t_{i-1}$ is the time interval between the $(i-1)$th and i th software failures [31]. The proposed model is used for predicting by taking execution time t_i as input and the accumulative number of failures N_i as output.

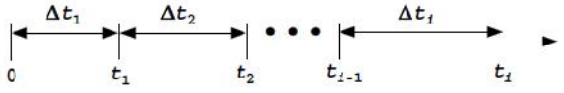

Fig. 3 Software failure process.

6.2 The proposed model

The prediction model based on neural network is shown in Fig.4, the input of the model is t_i which is encoded value of the execution time between (0,1), the output of the system is the cumulative number of failure N_i , the prediction model consist of three neural network combined together, each one is a three - layer feedforward neural network(FFNN) with n number of hidden neurons in the hidden layer, and there is a bias node in the input and the hidden layer, the activation function of the first neural network represent GO model, the activation function of the second neural network represent logistic curve model and the third neural network represent S-Shape model. For example the output of the first neural network as Eq. (11).

$$y(t) = 1 - e^{-\left((w_{21}(k1)+w_{22}(k2)+W_{23}(k3))+B*bw_{24}\right)} \quad (11)$$

$$k1 = 1 - e^{-w_{11} t + B * b\, w_{11}}$$
$$k2 = 1 - e^{-w_{12} t + B * b\, w_{12}}$$
$$k3 = 1 - e^{-w_{13} t + B * b\, w_{13}}$$

the output of the three neural network are combined together using mean value rule which is defined as Eq. (12).

$$N_i' = \frac{1}{M} \sum_{m=1}^{M} N_{i,m} \quad (12)$$

Where $N_{i,m}$ is the output of 3 neural network, m=1,2,…M.

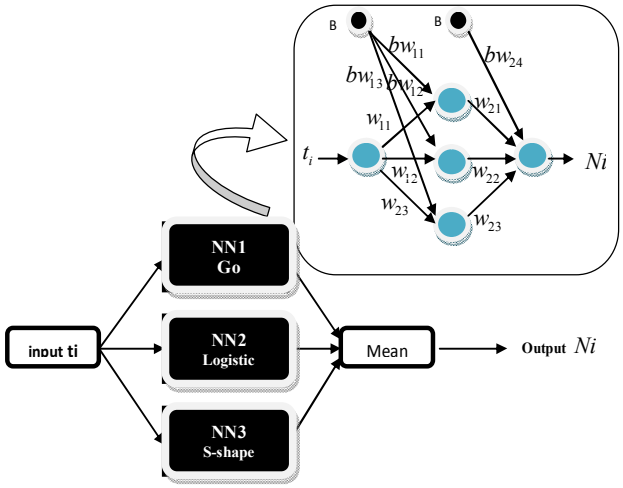

Fig.4 The architecture of the proposed model

6.3 The optimal set of PSO parameters

The PSO algorithm with well-selected parameter set can have good performance, In our software reliability prediction model, we use n number of swarms, the number of particle in each swarm is 25, the inertia weight is 0.9. according to Eberhart and Shi [10], the acceleration coefficients c1,c2 represent the stochastic acceleration that pulls each particle toward Gbest and Pbest position. In our model the acceleration coefficients c1,c2 changed in each iteration, and decreasing from large value to a small value through each iteration to improve the performance of the PSO algorithm,c1Max=0.9, c1Min=0.1, c2Max=0.2, c2Min=0.1.

6.4 Artificial neural network training and selecting the best architecture for the proposed model

The connection weights of the network have to be adjusted through a learning algorithm based on the training data, each neural network is trained with different initial weights connecting the three layers, the three NN trained in parallel by using particle swarm optimization(PSO) algorithm as we mentioned in section 4.
The number of neurons in the hidden layer determines the network's learning capabilities and its selection is a key issue in optimal network structure design. For selecting the best architecture of the NN(optimal number of the hidden layer neurons), we first build n models of NN architecture, each one have a different number of hidden neurons in the hidden layer, according to this we initialize n number of swarms based on the number of the models of the NN architecture. Fig.5 shows the flowchart of PSO learning process and selecting the best architecture. At the end of the training process (the maximum number of iterations), we get n number of Gbest (the optimal weights) of each swarm, then we evaluate the performance of each NN architecture based on the set of weights taken from Gbest, the evaluation is done by calculate the mean squared error(MSE), and the simulation results for the optimal number of the neurons in the hidden layer based on the minimum MSE.

7. Experiments

The proposed approach is applied to three software reliability datasets DS1[20], DS2[9] and DS3[23]. The DS1 and DS3 was collected from a real-time command and control application with 136 failures for DS1, 481 failures for DS3, the DS2 was collected from Operating System with 375 failures. The execution time and the number of failures of each dataset are normalized to the range of [0,1]. The model trained using 70% of the failure data for each dataset and the remaining data were used to test the model, the evaluation criteria (AE, RMSE) of the training and testing are shown in table 1. Figures 6 to 11 are showing the training and testing result for various datasets using our proposed model.

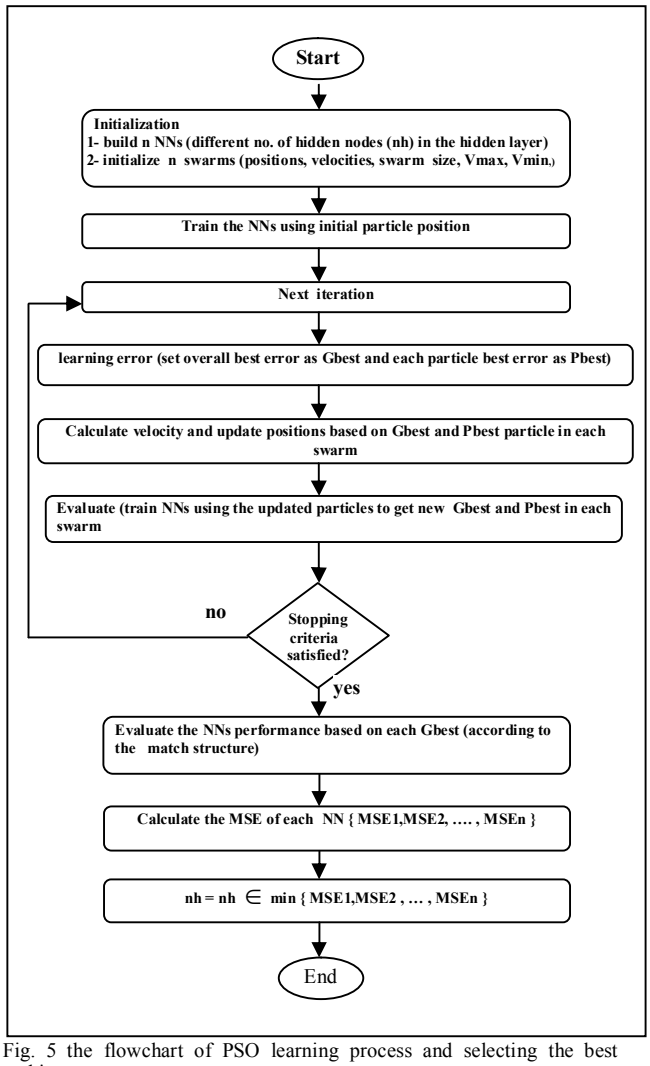

Fig. 5 the flowchart of PSO learning process and selecting the best architecture.

Table 1: results for the AE and RMSE of the training and testing dataset obtained using the proposed model

Dataset no.	*Best no. of hidden neuron*	*Training data*		*Testing data*	
		AE	*RMSE*	*AE*	*RMSE*
DS1	**15**	**5.338824**	**0.016248**	**7.178521**	**0.018374**
DS2	**15**	**5.666911**	**0.016613**	**6.087315**	**0.018605**
DS3	**3**	**3.219852**	**0.013820**	**4.481290**	**0.018025**

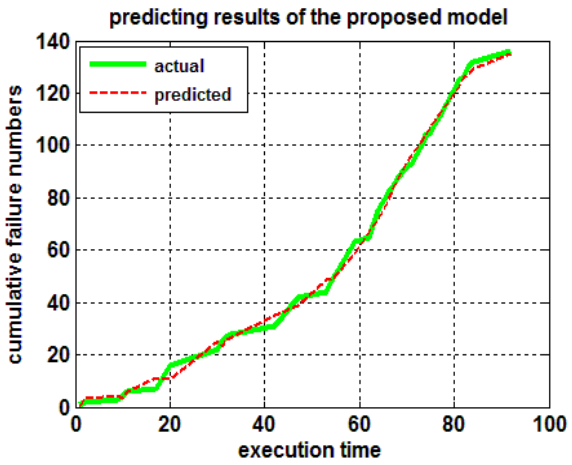

Fig. 6 Actual and predicted failure using DS1 (Training case)

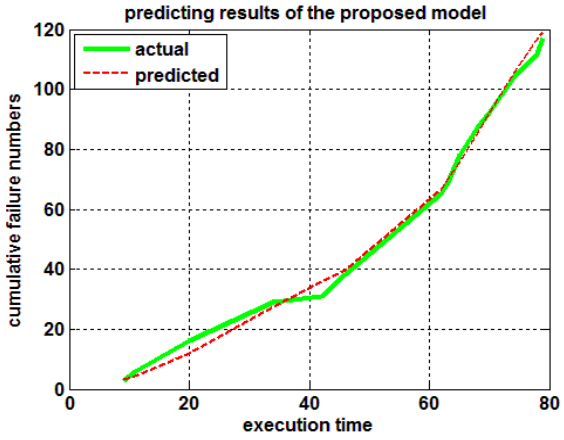

Fig. 7 Actual and predicted failure using DS1 (Testing case)

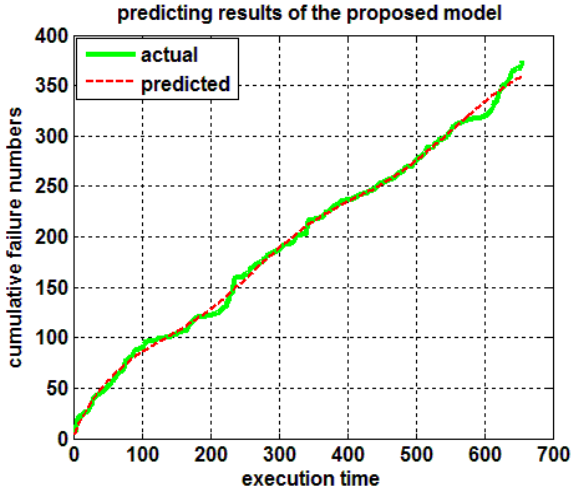

Fig. 8 Actual and predicted failure using DS2 (Training case)

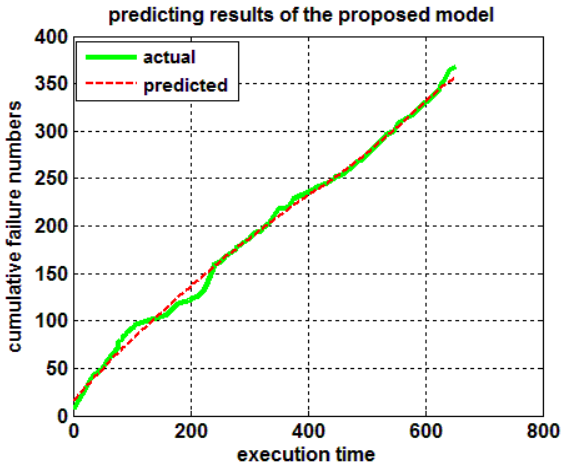

Fig. 9 Actual and predicted failure using DS2 (Testing case)

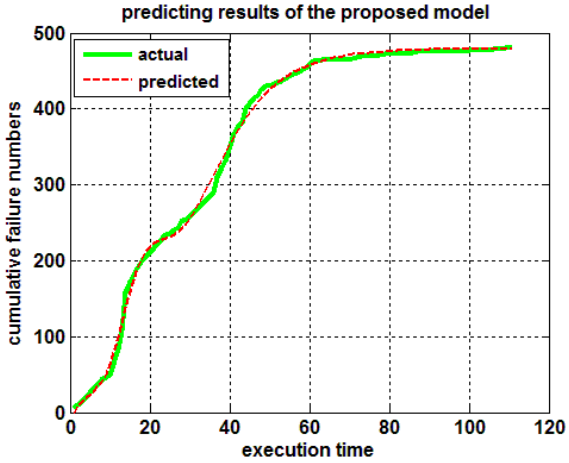

Fig. 10 Actual and predicted failure using DS3 (Training case)

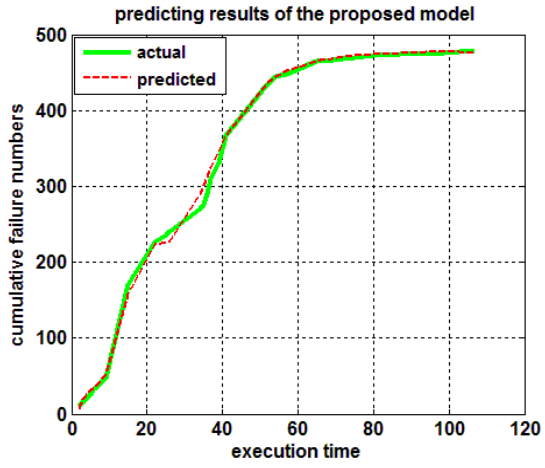

Fig. 11 Actual and predicted failure using DS3 (Testing case)

8. Conclusions

Software reliability is generally accepted as the major factor in software quality since it quantifies software failures. Neural networks trained by particle swarm optimization (PSO) has shown to be an effective non-parametric technique for software reliability prediction by optimizing the mean squared error(MSE), Selecting the best architecture of the network are also concerned for enhancing the performance of our model. The experimental results show that the proposed model gives acceptable result for different datasets relating to the prediction of the software cumulative failure.

References

[1] Abdull hamed H. N., Shamsuddin S. M., and Salim N. , " Particle Swarm Optimization for Neural Network Learning Enhancement" ,Jurnal Teknologi, 2008, pp.13 – 26.

[2] Aljahdali S. H., Buragga K. A., "Employing four ANNs Paradigms for Software Reliability Prediction: an Analytical Study", ICGST International Journal on Artificial Inteligence and Machine Learning, Vol. 8 , 2008, pp. 1- 8.

[3] Almering V., Genuchten M. V. , and Cloudt G., "Using Software Reliability Growth Models in Practice " IEEE computer society, 2007, pp. 82 – 88.

[4] Al-Rahamneh Z., Reyalat M. Sheta A. F., Bani-Ahmad S., and Al-Oqeili S., "A New Software Reliability Growth Model: Genetic-Programming-Based Approach" , Journal of Software Engineering and Applications, 2011, pp. 476-481.

[5] Benaddy M., Wakrim M., and Aljahdali S., " evolutionary neural network prediction for cumulative failure modeling", IEEE., 2009, pp. 179–184.

[6] Bisi M. , and Goyal N. K., " Software Reliability Prediction using Neural Network with Encoded Input ", International Journal of Computer Applications , vol. 47, No. 22 , 2012, pp. 46 – 52.

[7] Blum C. ,and Merkle D., Swarm Intelligence Introduction and Applications, Springer-Verlag Berlin Heidelberg., 2008.

[8] Cai K.Y., Cai L. , Wang W. D., Yu Z. Y. ,and Zhang D., " On the neural network approach in software reliability modeling" , The Journal of Systems and Software, vol. 58, 2001, pp. 47- 62.

[9] A. Data & Analysis Centre for Software DACS , https://www.thedacs.com/databases/sled.

[10] Eberhart R.C.,and Shi Y.," Particle Swarm Optimization : Developments, Applications and Resources" , IEEE, 2001 , pp. 81 – 86.

[11] Haque F., and Bansal, S. "Software Reliability Estimation Models: A Comparative Analysis" , International Journal of Computer Applications, Vol. 43, 2012, pp. 27 – 31.

[12] Hu Q.P., Xie M. and Ng S. H., " Early Software Reliability Prediction with ANN Models" ,IEEE, 2006, pp. 210-220 .

[13] Huang C. Y.,and Lyu M. R., "Estimation and Analysis of Some Generalized Multiple Change-Point Software Reliability Models" ,IEEE Transactions on Reliability, Vol. 60, 2011,pp. 498- 514.

[14] Inoue S., and Yamada S."Two-Dimensional Software Reliability Measurement Technologies", IEEE , 2009, pp. 223 – 227.

[15] Kapur P. K., Yadavalli V. S. S., Khatri S. K. , and Basirzadeh M. " Enhancing Software Reliabilityof a Complex Software System Architecture Using Artificial Neural-Networks Ensemble" , International Journal of Reliability, Quality and Safety Engineering, Vol. 18, 2011, pp. 271–284.

[16] Karunanithi N. , Whitley A., Malaiy Y. K. ," Using Neural Networks in Reliability Prediction" ,IEEE , 1992, pp. 53 – 59.

[17] Karunanithi N. ,and Malaiya Y. K., "The Scaling Problem in Neural Networks for Software Reliability Prediction" , IEEE , 1992, pp. 76 – 82.

[18] Karunanithi N., Malaiya Y. K .,and Whitley D. ," prediction of software reliability using neural networks" , IEEE, 1991, pp. 124 – 130.

[19] Karunanithi N., Whitley D., and Malaiy Y. K. ," Prediction of Software Reliability Using Connectionist Models" , IEEE Transactions on Software Engineering, Vol. 18, 1992, pp. 563 – 574.

[20] Lyu M. R., Handbook of software reliability engineering . New York: McGraw-Hill, 1996.

[21] Quadri S. M., Ahmad N. ,and Farooq S. U. " Software Reliability Growth Modeling With Generalized Exponential Testing –Effort And Optimal Software Release Policy" , Global Journal of Computer Science and Technology , 2011, pp.27 – 42.

[22] Sharma K., Garg R., Nagpal C. K., and Garg R. K., " Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach" , IEEE TRANSACTIONS ON RELIABILITY, Vol. 59, 2010, pp. 266 – 276 .

[23] Sheta A., "Reliability Growth Modeling for Software Fault Detection Using Particle Swarm Optimization" , IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, 2006, pp. 3071 - 3078 .

[24] Singh Y. ,and Kumar P., "Prediction of Software Reliability Using Feed Forward Neural Networks", IEEE, 2010, pp. 1- 5.

[25] Sitte R.," Comparision of software–reliability-growth predictions:neural networks vs parametric recalibration," IEEE Transactions on Reliability, Vol. 48, 1999, pp. 285-291.

[26] Su Y., Huang C., Chen Y., and Chen J. , "An Artificial Neural- Network-Based Approach to Software Reliability Assessment" , IEEE, 2005, pp. 1- 6.

[27] Su Y.,and Huang C. , " Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models" ,Elsevier Inc. The Journal of Systems and Software, 2007, pp. 606 – 615.

[28] sumathi S. ,and surekha p. Computational Intelligence Paradigms Theory and Applications using MATLAB .Taylor and Francis Group, LLC., 2010.

[29] Taylor B. J., Methods and Procedures For the Verification and Validation of Artificial Neural Networks, Springer Science + Business Media, Inc., 2006.

[30] Wang G. ,and Li W., " Research of Software Reliability Combination Model Based on neural Net" , IEEE, Second WRI World Congress on Software Engineering, 2010, pp. 253-256..

[31] Zheng J. ,"Predicting software reliability with neural network ensembles" , Elsevier Ltd, 2009, pp. 2116 - 2122.