Bulletin de la Société Royale des Sciences de Liège, Vol. 85, 2016, *p.* 187 - 202

A New Method to Predict the Software Fault Using Improved Genetic Algorithm

Fahimeh Sadat FAZEL [1]

Abstract

Maintaining and repairing the engineering machinery will be optimal when identification of defect or troubleshooting process, error analysis and fix it and in other word repair process, perform in highest precise, low time and cost. However speed and precise in troubleshooting process, defect analysis and repairing cause to decrease costs. So, the repair and troubleshooting role in a repair and maintenance system is so significant. While today's, the software has a main role to perform systems tasks, so they have a high importance in systems reliabilities. Therefore, to increase reliability, it is necessary to design systems against error if tolerable. With regard to increasingly development and applying software in different domains, software reliability has an important role during software life time. One of the most important solutions to increase software reliability is predicting software errors that cause to decrease in software maintenance cost in the future. There are many ways to predict software errors. Due to intelligence algorithms such as genetics algorithms, have a high ability to predict, so we can use them to predict software future condition or predict software errors. In this paper, we present a method to predict software error via genetics algorithm. The aim of this method is to predict software error with a higher precise and speed, and also to present structure that be implementation and expansion easily. The attained results show a desired performance of this method from time period for predicting error and output rate or recognition. The results show the recognition rates of suggested method more that 95 percent in best condition.

Keywords : error tolerance, reliability, error prediction, genetic algorithm.

1. Introduction

Given the important role of software systems, the quality of software components and their reliability is crucial. One of the important aspects of quality is reliability, thus, Software Reliability Engineering (SRE) is of great importance. One of the important aspects of SRE is software reliability modeling. Software reliability modeling became popular and prevalent from the early 1970s. Accordingly, the original method is modeling of previous erroneous data to predict its behavior in the future. This method uses the number of errors observed in each period of time or the time between the occurrences of software errors. Error Tolerance refers to the capability of the system which enables the system so even when an error occurs, it can cover the error and prevent system failure. This capability in many systems is among the most important non-functional requirements of the system. For many systems, correct and reliable software operation is an important requirement, such as aerospace applications, air

[1] Fahimehsadatfazel65@gmail.com

traffic control, medical equipments, nuclear, and electronic banking. The cost and failure of these systems can result in a broad range of human injury, financial loss, etc. Given that, today, software plays the major role in carrying out systems' functions, hence, it is of special importance on the reliability of systems. Therefore, to increase reliability, it is necessary to design error-tolerant systems. Software error tolerance is often obtained by redundancy and diversity which increases the complexity of software design. Due to the increasing development and application of software in various areas, software reliability plays an important role throughout the software life. Software reliability is of the main features of software quality and the most important factor that customers expect from software products. The software reliability declines by software failure at run time, and the degree of error tolerance, software maturity, and the software recovery capability are important and affecting factors on software reliability. There are different models for estimating software reliability. One of the solutions to enhance software reliability is forecasting software errors, reducing the cost of software maintenance in the future. There are many methods for forecasting software error. Considering the fact that intelligent algorithms such as genetic algorithms have a high ability to forecast, they could be used in forecasting the future status of software or software error.

2. Theoretical foundations and background

There are important definitions regarding the structure of reliability which are briefly described here.

Fault: the assumed or detected cause of error, sometimes also called Bug

Error: a part of the system status that leads to a failure. Error may not be detected (secret) or may be detected. Error may diffuse, i.e. generate other errors. When errors are detected, it means that there are faults.

Failure: failure occurs when the provided service is deviated by the system from its main function, or otherwise, generates wrong result. So, through fault tolerance in software, failures may be prevented.

In this diagram, the error rate per unit time is specified. There are two major differences between the hardware and software curves. The first difference is that the hardware does not have increasing failure till the final stage; however, the software, in the final stage, gets closer to obsolescence, and there is no incentive for any upgrade or change in software. Thus, the failure rate will not change. The second difference is that in the useful or used stage in the software life cycle, each time, a sharp rise in the failure rate upgrades the software. The failure rate after each upgrade remains constant in a time limit.

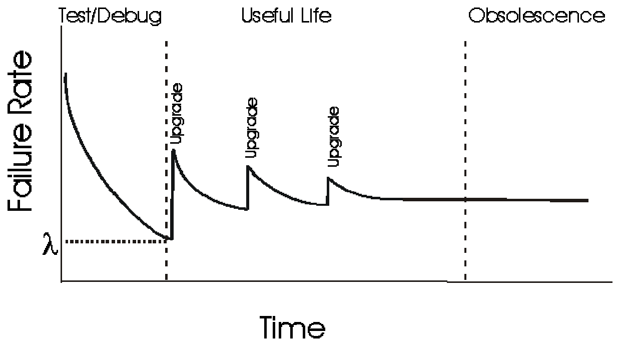

Figure 1: Software lifecycle and its stages

Usually in reliability growth models, the possibility of error over a period of time [0, T] is used as software reliability. Formula 1 shows the formula describing software reliability:

$$R(T) = \Pr ob\{no\_failure\_in[0,T]\} \quad (1)$$

Today, the test stage is considered one of the most important stages of software development. As application programs are rapidly developing in different environments, there is a high focus on software testing. In the test phase, the part of the code that may have the greatest number of faults, more focus is placed to eliminate the complexity of this part, hence, reducing the development costs and upgrade the software lifetime. But, it needs to predict the location of potential faults. In 1995, Scherrer did research on error forecast in software. In his study, neural networks were used to forecast fault. He tested his proposed structure on several applications of National Aeronautics and Space Administration. The results showed that the proposed method diagnosed error-prone modules, and through measures after the forecast, the detected module was assessed. A review of studies includes the following:

- Error forecast with the detailed structure
- Error forecast using detailed processing
- Error forecast using event processing
- Error forecast using fault tree structure
- Error forecast using statistical analyses
- Error forecast using intelligent algorithms

3. Development of hypotheses and conceptual model

The proposed method consists of three steps:

- Selection of appropriate software database to use software indices
- Extraction of important features using genetic algorithm
- Software error forecast using genetic algorithm output

In order to forecast error in each application, first the software database must be determined and then extract the indices by the genetic algorithm to determine the probability of error of the application as the output of genetic algorithm.

Given that finding code indicators is a major factor for having an appropriate data source for error forecast in an application, first, the main indicators must be selected to forecast error. One of the most important data bases used in error forecast is NASA MDP 3 database. This database contains 13 data sets which contains information about different understudy software applications at NASA database. This database is created through various software feedbacks from NASA, and the used indicators are appropriate criteria to determine the characteristics of the application code. Table 1 shows the data set of each system separately. Each row of this table represents a function set responsible for a specific performance in leading spacecrafts or satellites.

Table 1: NASA MDP data set and the related software system

Data set	Software system
CM1	Satellite instrument control system while flying or landing
JM1	Real-time forecasting system on Earth
KC1	Storage management system for receiving and processing data from the Earth
KC3	Management and storage of data sent to Earth
KC4	Shared server system with the Earth
MC1	Space shuttle combustion control software
MC2	Video control system for all parts of the satellite
MW1	No gravity control system
PC1	Flight control software and number one rotation around the Earth
PC2	Flight control software and number two rotation around the Earth
PC3	Flight control software and number three rotation around the Earth
PC4	Satellite angle control software while flying or landing
PC5	Cockpit security increase system

Table 2 lists the name of each set, the number of modules, and the error modules.

Table 2: Systems available in NASA MDP database and the number of modules and error modules

Dataset	Number of modules	Percentage of error modules
CM1	505	9.5
JM1	10878	19.3
KC1	2107	15.4
KC3	458	9.3
KC4	125	48.8
MC1	9466	0.7
MC2	161	32.29
MW1	403	7.69
PC1	1109	6.94
PC2	5589	0.4
PC3	1563	10.2
PC4	1548	12.2
PC5	17186	3

One of the most important sections of the proposed method is selection of criteria or important code indices to forecast the error speedily and accurately in the software. Figure 2 shows the method structure of a genetic algorithm used in the thesis as a flowchart.

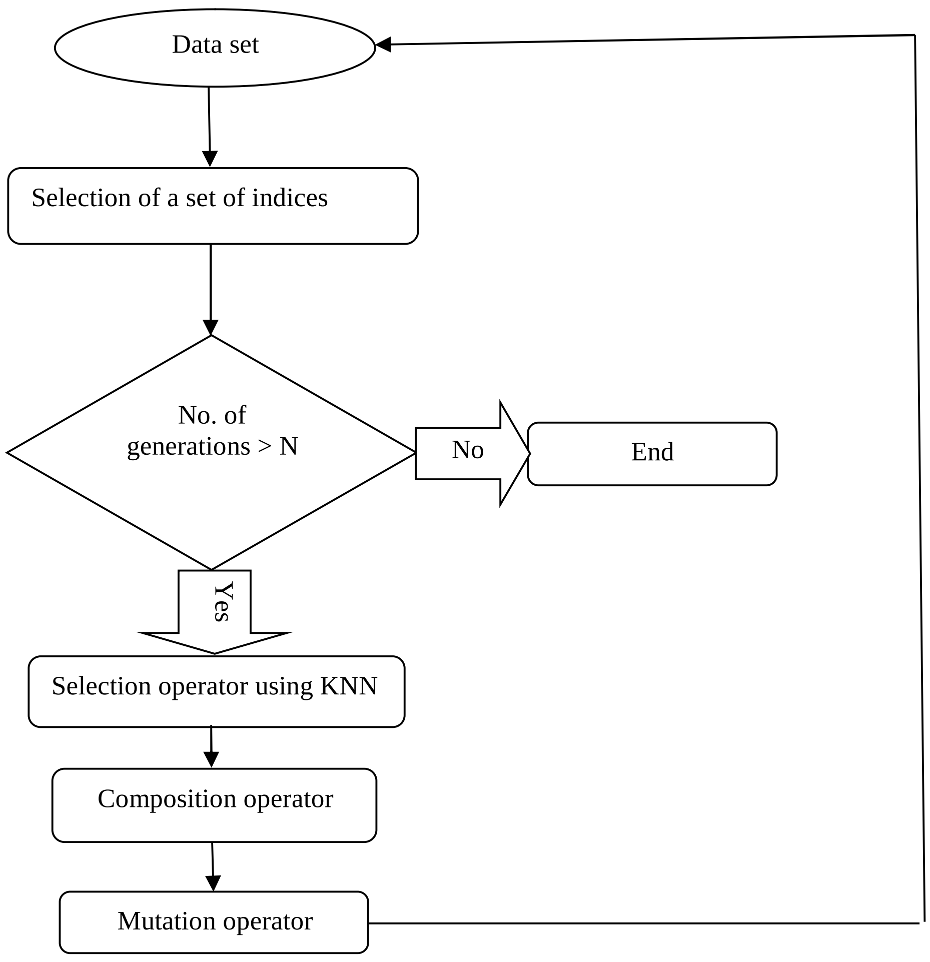

Figure 2: The structure of the genetic algorithm performance in selecting important indicators

Table 3: Structure of indices set in a system

Module No.	Index 1	Index 2	**....**	Index n	Problem in module

Given that the index table in each column is indicative of a function as Table 3, so for population genetic algorithm, bit array is used, i.e. each chromosome is composed of a number of ones and zeros. The number of genes per chromosome is dependent on the rate of system table indices (Table 1). If a gene is one, the index related to the gene is selected and considered in the operator. And if it is zero, it means that the index is ignored. Selection of the indices is done randomly. KNN algorithm is used to calculate the value of the indices. KNN is a supervised training algorithm.

In general, this algorithm is used for two purposes: to estimate the function of density distribution of training data and for classification of test data based on training patterns. KNN is the easiest and most popular method based on learning sample. The assumption is that all samples are points in real n-dimensional space, and the neighbors are determined based on the standard Euclidean distances. *k* is the number of neighbors. To measure the performance of KNN function, the error rate is calculated using regression. In fact, lower the resulting error rate, better the function output. The formula for calculating the error rate is obtained using the regression is presented as equation (2).

$$S = \sum_{i-1}^{n} |y_i - f(x_i)|. \qquad (2)$$

Y_i is the real output (main class), and $f(x_i)$ is the calculated output by KNN (KNN classification). In the proposed method, the composition operator is from the two point random method, and the mutation operator on *A*% is from the new generation. Equations 3 and 4 are composition and mutation operators, respectively.

t = Rand ((length-1)*cr/2) (3)

t = Rand ((length-1)/2) (4)

t is considered the first point, and length-t is the second point. *cr* is considered an optional parameter to limit or broaden the random process of generation of children.

4. Methodology

This method works based on the number of correct or incorrect answers. The method is based on using confusion matrix. Confusion matrix shows the performance of the relevant algorithms. Confusion matrix for the proposed method is presented as Table 4.

Table 4: Profile of indices' structure in a system

Error in module		Reality	
		Positive	Negative
Forecast	Positive	True positives (TP)	False-positives (FP)
	Negative	False- negatives (FN)	True-negatives (TN)

The accuracy of the performance is calculated as equation 5:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \tag{5}$$

In the proposed method, one of the most important factors to decide about the genetic algorithm is using appropriate data source. The dataset used in this method involves the number of variables, operators, operands used, rings, modules, and conditional structures. Fitness function algorithm, using this dataset, modifies the population generated at each stage, and finally, all points leading to software fault are detected.

5. Data analysis and results

The results of detection simulation of software error are provided, using genetic algorithm and MATLAB software. K-nearest neighbor function is considered. To determine the performance of the proposed method, the following parameters have been changed in every simulation: dataset, the number of genetic algorithm generation, the population requirement of education and testing, and changes in the composition operator

Simulation number one, variations in dataset

In this simulation, the initial population is 50, the number of generations is 100; the composition rate for the next generation is 0.8, the mutation operator 0.1, the data for training and selection of indices, 70%, and the test data rate is considered 30%. The simulation was performed with two datasets, including satellite landing equipment control software and the Earth data management software. The forecast results specify that if the frequency of the number of modules is higher at any rate, the index selection is done with higher accuracy.

Example: CM1 and PC1 data set

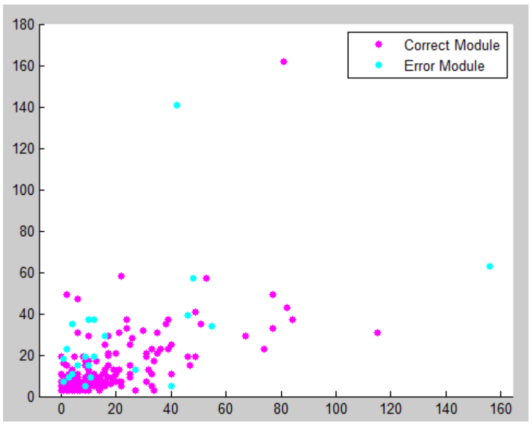

Figure 3: Detection of correct and error-prone modules in the data set of satellite equipment control software

The results show that the accuracy of the proposed method is 90.8 %.

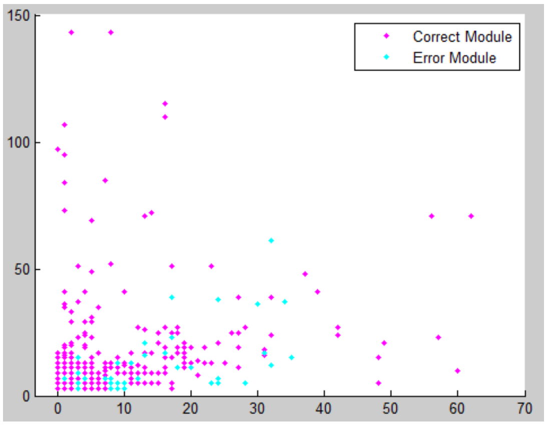

Figure 4: Detection of correct and error-prone modules in data management software from the Earth

The results of the performance accuracy of the proposed method show 96.16 %.

Simulations number two, variations in the number of generations

In this simulation, CM1 dataset with 37 indicators, initial population of 50, the composition rate for the next generation population 0.8, mutation operator rate of 0.1, the data for training and selection of indices, 70%, and 30% test data are used. The simulation was performed with the number of generation of 50 and 200. The results specifies the error forecast rate that the generation number should be selected empirically and according to the dataset, and practically, it generation increase or decrease will have no demonstrable effect on upgrading the performance of the method.

Example: CM1 data set with 50 and 200 generations

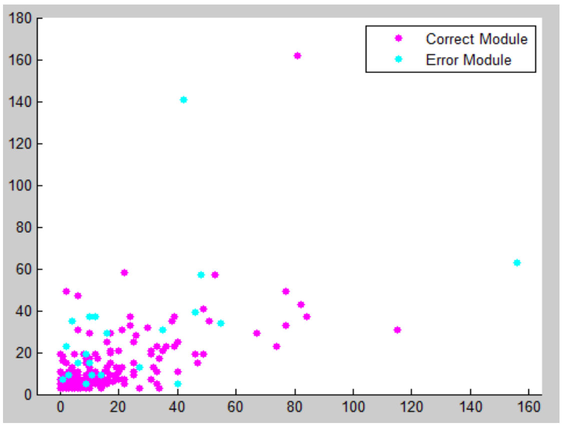

Figure 5: Detection of correct and error-prone modules in 50 generations

The results show that the accuracy of the proposed method is 91.17 %.

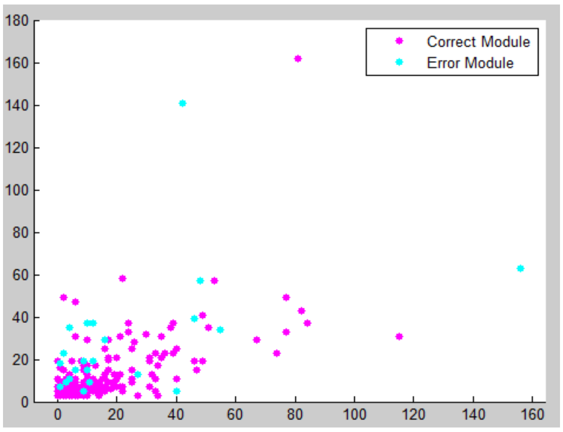

Figure 6: Detection of correct and error-prone modules in 200 generations

The results show that the performance accuracy of the proposed method is 90.08 %.

Simulations number three, variations in population requirement of training and testing

In this simulation, CM1 dataset with 37 indicators, initial population of 50, the number of generation of 100, composition rate of 0.8 for the next generation population, and mutation operator of 0.1 are used. At first, the simulation is carried out with 60% data rate for the training and selection of indices and the test data of 40%, and at the second time, 80% data rate for training and selection of indices, and test data of 20%. Error forecast results indicates that variations in population requirements of training and testing are also empirical, such as the previous indicator, which requires the best value empirically. By default, in a lot of research studies, the training data is considered 70% and the test data 30%.

Example: CM1 dataset with 60 % population for training and 40% for testing and with 80 % for training and 20 % for testing

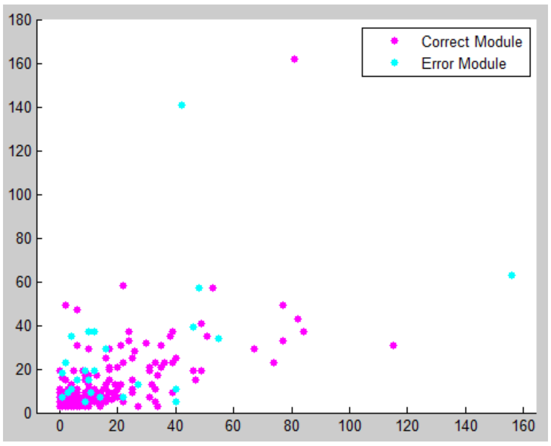

Figure 7: Detection of correct and error-prone modules with 60% population for training and 40% for test

The results show that the accuracy of the proposed method is 90.10 %.

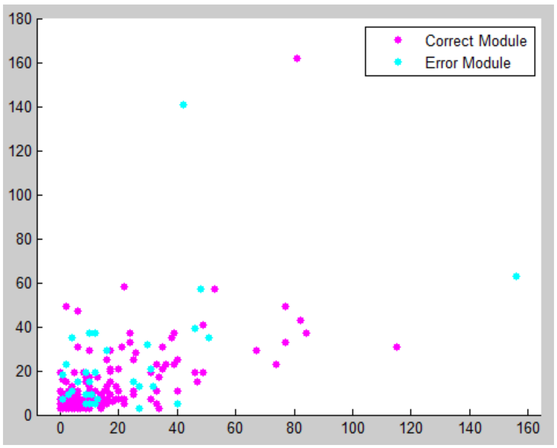

Figure 8: Detection of correct and error-prone modules with 80% population for training and 20% for test

The results show that the accuracy of the proposed method is 94.27 %.

Simulation number four, variations in the composition operator rate

In this simulation, CM1 dataset with 37 indicators, initial population of 50, 100 generations, 0.1 mutation operator, 70% training data and indices selection, and 30% test data rate are used. The effect of composition operator in the first simulation is 0.5 and in the next simulation is selected 0.9. Although composition operator is one of the most effective factors on genetic algorithm, the results of error forecast indicates that the composition operator changes have no effect on the final result, and after a certain number of generations in this method, the same results are obtained.

Example: CM1 dataset and composition operator equal to 0.5 and composition operator equal to 0.9

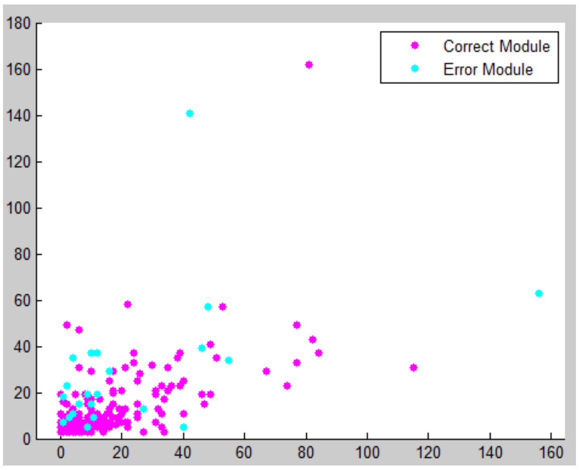

Figure 9: Detection of correct and error-prone modules and operator composition equal to 0.5

The results show that the performance accuracy of the proposed method is 90.80 %.

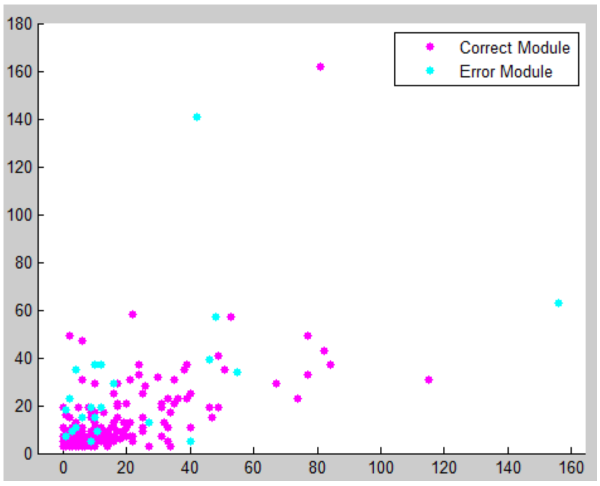

Figure 10: Detection of correct and error-prone modules and composition operator equal to 0.9

The results show that the performance accuracy of the proposed method is 90.80 %.

6. Conclusion

Comparison of the proposed approach with other methods

So far, various algorithms have been proposed and implemented to forecast software errors. The most important algorithms are methods based on statistics, machine learning techniques, neural networks, and cluster-based methods, and more recently, parametric models' methods, nonlinear time series analysis, and data mining. Almost, all these methods have the following common faults: because of the functionality limit of a software and creation of a matrix from different running forecasted or non-forecasted statuses during using application and, most importantly, the constant change of constituent elements of the matrices' components at larger time intervals during actual software running, virtually, there is no possibility of superficial and deep navigation of all software sectors. Accordingly, this process will require a lot of time even if all sectors and different functional statuses of software are navigated. Above all, these algorithms can usually provide information only on the current software reliability at proper run of processes in real time, or eventually proper and errorless run of later processes. The problem is that, in general, an application is so complicated that these methods cannot ever guarantee software reliability in future performances. Moreover, these algorithms cannot review statuses and parts of a running application when an error occurs, thus understand the root causes of errors and failures. Since the search space of the so-called problem is very large, genetic algorithm is a perfect solution for issues with large search space and can accurately and more quickly find the location of the error. A new method was presented for predicting error using genetic algorithm. The results show that the proposed method's optimal performance is considered in terms of error forecast duration, the output rate, or its detection. The results, at best, suggest a higher detection rate of the proposed method higher than 95%.

Recommendations

- Using new intelligent algorithms

As discussed in the third section, the proposed method uses the genetic algorithm. Given that the genetic algorithm is one of the oldest optimization algorithms, it is recommended to use newer algorithms such as flying bats or cuckoo search algorithms.

- The use of other datasets

Although one of the most significant datasets to forecast error or software reliability is NASA MDP and it is used in many research studies; however, new datasets have recently been proposed. It is suggested to review and discuss the presented method in this thesis with respect to these datasets.

- Use of self-organizing fuzzy neural network

This is one of the most notable structures to categorize and identify true or false grouping of a fuzzy neural network set. If the fuzzy neural network can change its layers during training, it is self-organizing. One of the strongest and fastest forecast structures based on early training is the self-organizing fuzzy neural network. It is suggested to use this structure to forecast software error in future researches.

References

1. Ak R, Li Y, Vitelli V, Zio E. Multi-objective genetic algorithm optimization of a neural network for estimating wind speed prediction intervals. Chair on Systems Science and the Energetic Challenge, Department of Energy, 1, 2013.
2. Aljahdali S, Telbany M. Software reliability prediction using multi objective genetic algorithm. Computer Sciences Department, Al-Taif University, 978, 2009, 293-300.
3. Bhattacharya S, Rungta S, Kar N. Software fault prediction using fuzzy clustering & genetic algorithm. International Journal of Digital Application & Contemporary Research, 2(5), 2013.

4. Binkley D, Feild H, Lawrie D, Pighin M. Software fault prediction using language processing. Loyola College, Universita’ degli Studi di Udine, 2009.
5. Boetticher G. Nearest neighbor sampling for better defect prediction. University of Houston – Clear Lake, 2005.
6. Catal C. Performance evaluation metrics for software fault prediction studies. Department of Computer Engineering, 9(2), 2012, 193-206.
7. Chang R, Mu X, Zhang L. Software defect prediction using non-negative matrix factorization. Xi’an Research Inst. of Hi-Tech, Xi’an, China, 6(11), 2011, 2114-2120.
8. Goyal R, Chandra P, Singh Y. Suitability of KNN regression in the development of interaction based software fault prediction models. International Conference on Future Software Engineering and Multimedia Engineering, 6, 2014, 15-21.
9. Jiang Y, Cukic B, Menzies T, Bartlow N. Comparing design and code metrics for software quality prediction. The Lane Department of Computer Science and Electrical Engineering, 978, 2008.
10. Kaur A, Gulati S. A framework for analyzing software quality using hierarchical clustering. International Journal on Computer Science and Engineering, 3(2), 2011, 854-861.
11. Li H, Lu M, Zeng M, Huang B. A non-parametric software reliability modeling approach by using gene expression programming. Journal of Information Science and Engineering, 28, 2012, 1145-1160.
12. Meenakshi P, Meenu S, Mithra M, Leela P. Fault prediction using quad tree and expectation maximization algorithm. International Journal of Applied Information Systems, 2(4), 2012, 36-40.
13. Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering, 32(11), 2007, 1-12.
14. Paksoy A, Göktürk M. Information fusion with dempster-shafer evidence theory for software defect prediction. Procedia Computer Science, 3, 2011, 600-605.
15. Rodríguez D, Ruiz R, Riquelme J, Ruiz J. Searching for rules to detect defective modules: A subgroup discovery approach. Department of Computer Science, University of Alcalá, School of Engineering, Pablo de Olavide University, Department of Computer Science, University of Seville, 2011, 1-17.
16. Sandhu P, Khullar S, Singh S, Bains S, Kaur M, Singh G. A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm. World Academy of Science, Engineering and Technology, 48, 2010, 562-567.

17. Schroter A, Zimmermann T, Zeller A. Predicting component failures at design time. Saarland University, 1, 2006.
18. Shepperd M, Song Q, Sun Z, Mair C. Data quality: Some comments on the NASA software defect data sets, 2012, 1-13.
19. Singh P, Verma S. An efficient software fault prediction model using cluster based classification. International Journal of Applied Information Systems, 7(3), 2014, 35-41.
20. Wahono R, Herman N. Genetic feature selection for software defect prediction. American Scientific Publishers, 20, 2014, 239-244.
21. Wahono R, Suryana N, Ahmad S. Metaheuristic optimization based feature selection for software defect prediction. Journal of Software, 9(5), 2014, 1324-1333.