2020-02

Improving Software Reliability Growth Models Based on Neuro-Genetic Approach

Chalachew, Abraham

BAHIR DAR UNIVERSITY

BAHIR DAR INSTITUTE OF TECHNOLOGY

SCHOOL OF RESEARCH AND POSTGRADUATE STUDIES

FACULTY OF COMPUTING

Improving Software Reliability Growth Models Based on Neuro-Genetic Approach

MSc. Final Thesis

By

Abraham Chalachew

Program: Software Engineering

Supervisor: Dr. Mekuanint Agegnehu

February 2020

Bahir Dar, Ethiopia

Improving Software Reliability Growth Models Based on Neuro-Genetic Approach

Abraham Chalachew

Improving Software Reliability Growth Models Based on Neuro-Genetic Approach submitted to the school of Research and Graduate Studies of Bahir Dar Institute of Technology, BDU in partial fulfillment of the requirements for the degree of MSC in Software Engineering in Faculty of Computing.

Advisor Name: Dr. Mekuanint Agegnehu

Bahir Dar, Ethiopia

February 26, 2020

DECLARATION

I declare that all the work contained in this thesis is my own work unless acknowledged. All of my work has not been previously summited for the academic degree. All source of the information has been acknowledged by means of appropriate references.

Name of student Abraham Chalachew Signature

Submission date: 17/06/2012

Place: Bahir Dar

The thesis has been submitted for approval as advisor of the university.

Advisor Name: Mekuriaw A. (PhD)

Advisor's Signature:

Bahir Dar University

Bahir Dar Institute of Technology-

School of Research and Graduate Studies

Computing Faculty

THESIS APPROVAL SHEET

Student:

Abraham Chanie — Signature — 17/06/2012

Name — Signature — Date

This is to certify that the work of our thesis carried out for partial fulfillment of the thesis requirements for the award of the Degree of MSC in Software Engineering.

Approved by:

Advisor:

Mesfin A (PhD) — Signature — 17/06/2012

Name — Signature — Date

External Examiner:

Ayalew Belay Habtie (PhD) — 17/06/2012

Name — Signature — Date

Internal Examiner:

Esubalew Alemneh (PhD) — 18-Feb-2020

Name — Signature — Date

Chair Holder:

Gebeyehu B (Dr) — 17/06/2012

Name — Signature — Date

Faculty Dean:

Belete B. — Signature — Date

Name

To my father and mother

Acknowledgement

First and for most I would like to thanks GOD he has helped me in every aspect of my life. Next to that I would like to express my sincere gratitude to my Advisor **Dr. Mekuanint Agegnehu** for his Advice and comment in all of my paper. I appreciate and value his esteemed direction and support from the beginning of our work to the end of our works. I would also like to express my thanks to all who extended unlimited help to me during my work.

Abstract

Software engineering is incomplete without Software reliability prediction. For characterizing any software product quality quantitatively during phase of testing, the most important factor is software reliability assessment. Many software reliability growth models (SRGM) which is used for predicting in software reliability, however, no single model can give accurate prediction. For this the Artificial Neural Network (ANN) based software reliability model is introduced. In this thesis ANN based software reliability models for better reliability prediction in a real case is described and the growth of software reliability using ANN based model is presented. We proposed a neuro-genetic approach for the ANN based software reliability model by optimize the weights of the network by using proposed genetic algorithm (GA). Training the ANN using Back-propagation algorithm (BPA) to predict the software reliability is the first action. Than train our model global optimize the weight of the networks by using the proposed GA. Using two datasets contain cumulative executive time and cumulative no of software failures are applied to the proposed models. These datasets are obtained from software projects. Then it is observed that the results obtained indicate a significant improvement in performance by using genetic algorithm in ANN based software reliability models over the normal algorithm of ANN based software reliability models. Numerical and graphical explanations show that proposed model for software reliability prediction since its fitting and prediction error is much less relative to the normal algorithm of ANN based software reliability model.

Table of Contents

DECLARATION ..... **Error! Bookmark not defined.**
Acknowledgement vi
Abstract vii
CHAPTER ONE 1
Introduction 1
1.1 Background 3
1.2 Statements of the problem 4
1.3 Objectives 5
1.3.1 General objective 5
1.3.2 Specific Objective 5
1.4 Scope of the study 5
1.5 Methods 6
1.5.1 Problem identification and motivation 6
1.5.2 Objectives for a solution 6
1.5.3 Design and development 6
1.5.4 Demonstration 7
1.5.5 Evaluation 7
1.5.6 Communication 7
1.6 Significance of the study 7
CHAPTER TWO 9
Literature Review 9
2.1 Software Reliability Growth Model (SRGM) 9
2.2 Overview of The Artificial Neural Networks 10
2.2.1 Artificial neural networks (ANNs) 10
2.3 Genetic Algorithm 13
2.4 Literature Review 15
CHAPTER THREE 18
Methodology 18
3. Introduction 18

3.1 Software failure data sets 20
3.2 Normalized of software failure data sets 20
3.3 Model Architecture 21
3.4 The neural networks combination model. 22
3.5 ANN Approaches for Software Reliability Modeling 23
3.5.1 The selection of the base models 23
3.6 Training neural network: Back propagation vs. Genetic Algorithms 25
CHAPTER FOUR 28
Result and Discussion 28
4.1 Implementation of the Proposed ANN based SRGM to predict Software reliability 28
4.2 GA implementation for training of ANN based model for software reliability prediction. 28
4.3 Different Performance Measures 29
CHAPTER FIVE 42
Conclusions and Recommendations 42
5.1 Conclusions 42
5.2 Recommendations 43
Reference 44
Appendix A 46
Datasets 46

List of Tables

Table 2. 1 SRGMs with mean value function 9

Table 3. 1 The mean value function of the selected models is the following. 24

Table 4. 1 Results for Backpropagation Algorithm 36
Table 4. 2 Results for Genetic Algorithm 36
Table 4. 3 comparasion of Backpropagation Algorithm and Genetic algorithm in DS1 36
Table 4. 4 comparasion of Backpropagation Algorithm and Genetic algorithm in DS2 36

List of Figures

Figure 2. 1 Neuron 11
Figure 2. 2 A multi-layer feedforward ANN 12

Figure 3. 1 Flowchart of the proposed model 19
Figure 3. 2 Architecture of the proposed model 21
Figure 3. 3 Neural network optimization by the genetic algorithm 26

Figure 4. 1 The snapshot of the performance measure values using MSE for Dataset1 30
Figure 4. 2 The snapshot of the performance measure values using RMSE for Dataset1 31
Figure 4. 3 The snapshot of the performance measure values using RMSE for Dataset2 32
Figure 4. 4 The snapshot of the performance measure values using MSE for Dataset2 33
Figure 4. 5 Comparison between Actual data and predicted data in DS1 34
Figure 4. 6 Comparison between Actual data and predicted data in DS2 35
Figure 4. 7 Results that shows the BPA in both datasets 38
Figure 4. 8 Results that shows GA in both datasets 39
Figure 4. 9 Comparison of backpropagation algorithm and genetic algorithm in DS1 40
Figure 4. 10 Comparison of backpropagation algorithm and genetic algorithm in DS2 41

Acronym

SRGM	Software Reliability Growth Model
NHPP	Non-Homogenous Poison Process
LSE	Least Square Estimation
MLE	Maximum Likelihood Estimation
ANSI	American National Standards Institution
ANN	Artificial Neural network
GA	Genetic Algorithm
BPA	Back propagation Algorithm
SRP	Software Reliability Prediction

CHAPTER ONE

Introduction

As (Mallikharjuna & Kodali, 2015) introduced modern society is highly engaged with the role of software. Software engineers and software development organizations seeks great responsibility on maintaining quality, reliability and customer satisfaction with the software product. (Andersson, 2007) introduced software development testing is generally considered as one of the major quality control techniques. In order to calculate and predict the product quality software reliability is found as a significant attribute (Mir, 2011). As (Ramasamy & Lakshmanan, 2016) introduced American National Standards Institution (ANSI) defines the software reliability as the probability of failure-free operation of a software system for a specified period of time in a specified environment. As (Ong, Isa, Jawawi, & HALIM, 2017) introduced the growth of software reliability increases through the removal of the faults during software failures in the test phase. Software reliability is one part of software quality, it is the highest concern by developers and project managers with the considerations with business profitability, user safety and preservation of the environment. Software reliability is an important for software development because unreliable software have some error or bugs that may causes software failures to occur if thus problem is not handled early. Examples of system failure has unfavorable impact on the environment, caused economic loss, even harmful to human lives (Ong et al., 2017). Thus software reliability prediction(SRP) has become crucial activity in software development process in order to produce reliable and good quality software. (Anjum, Haque, & Ahmad, 2013) presented among the prediction model, Software reliability growth model (SRGM) has been widely used in many software domains, such as telecommunication, embedded systems, military, banking and industrial control system. SRGM estimate the present reliability by estimating the model parameters and predict the future reliability of a system using practical failure data.

SRGM are generally classified into two categories: Parametric and Non-parametric models.
Parametric models estimate the model parameters based on the assumptions of underlying distributions. As (Su & Huang, 2007) introduced thus model depends on some assumption, it is believed that no single model can provide accurate estimation in all situations. As (Lakshmanan & Ramasamy, 2015) introduced one of the difficult tasks in parameter estimation of traditional SRGMs is estimating ranges and start values for each parameter to be estimated.

Thus models have certain assumptions regarding numerous factors such as software development process, software development organization, software use characteristics, nature of software faults and software complexity to predict software reliability. Some assumptions are not valid in real cases in software industry. Non-parametric models are used to overcome this problem. They do not consider any assumption about software development process and software development organization. They only used failure history of software to predict its reliability. The non-parametric models use artificial neural network (ANN), support vector machine (SVM), fuzzy logic (FL) and genetics algorithm (GA) to predict software reliability. Among several non-parametric models, **ANNs** are normally used to predict software reliability. The failure behavior of software failure data follows non-linear pattern between input and output. As (Lakshmanan & Ramasamy, 2015) introduced all soft computing techniques such as Artificial Neural Network, Fuzzy systems, Genetic algorithms are the non-parametric models .

The problems with parametric SRGM is solved by soft computing techniques. So there is no specify the range of values in advance for each parameter which is a complex task (Mallikharjuna Rao & Anuradha, 2016). However, traditional SRGM does not have determined parameter. Optimization of these parameter is necessary task; these parameters are determined by least square error(LSE). Such a software failure data may not satisfy such a distribution.

In this paper, we use GA to train our proposed model using failure data sets of the software by using global optimize the weight and parameter of the ANNs. It is a non-linear continuous function, ANNs are widely used in various fields of software engineering such as cumulative failure prediction, time between failure prediction, classification of software modules, software development effort prediction and software fault localization.

Most of the ANN based models available in literature are trained using back propagation algorithm. It is a gradient based algorithm which suffers from local optimum problem. Genetic algorithm technique is widely used to solve the local optimum problem, in this work, there are three novel aspects. First, a new ANN architecture is proposed to predict software reliability taking testing time as input of the model. The purpose is used to distribute the inputs of ANN in a specified range for better training of ANN. Second, the proposed ANN model is trained using Genetic algorithm to predict cumulative number of failures in software. Third, the proposed model is validating by using two data sets available in literature.

1.1 Background

Software Reliability

The probability that a software will perform a required function under stated conditions for a specified period of time is known as software reliability's. It is a very important factor to determine the qualities of software products during testing phase. The major benefits of software reliability measurement are planning and controlling the resources during software development process for developing high quality software. It gives confidence about software correctness. And additional cost is minimizing during testing and reliability should be improved. As (Bhuyan, Mohapatra, & Sethi, 2016) introduced at the time of development of any product or system like commercial, military or any other applications, we need to ensure its reliability and consistency in its performance, because a systems reliability has a major impacts of maintenance, repair costs, continuity of services and customer satisfaction. At the end, the project manager needs to ensure that the software is reliable enough to be released into the market.

(Lyu, 1996) described As per ANSI definition, software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. As (Lakshmanan & Ramasamy, 2015) introduced Software reliability models facilitate estimation of the present or future reliability of a system by estimating the parameters used in the models using software failure data at a given time. Parametric models estimate the model parameters based on the assumptions of underlying distributions. As (Lakshmanan & Ramasamy, 2015) introduced Parametric models can be further divided into three types: Non-Homogeneous Poisson Process (NHPP), Markovian models and Bayesian models .

Software Reliability Measurements

- **Failure Rate***:* It is the rate of occurrence of failures. It also represents number of failures in specified period of time.
- **Mean Times Between Failures (MTBF):** It is the mean value of time and failures.
- **Availability***:* The probability that an item is in operable state at any time is called availability. It accounts for repairs and down time.

Software Reliability Growth Models

It includes two types of models

- Parametric models
- Nonparametric models

Parametric models are based on non-homogeneous Poisson process (NHPP). ANN based software reliability model is non parametric model and based on statistical failure data. Non parametric models are more flexible.

1.2 Statements of the problem

In the modern age, it is a big challenge for software developers to quickly design, implement, test and maintain complex software systems. (Bhuyan, Mohapatra, & Sethi, 2014) introduced also it is difficult task for software companies to deliver good quality software in appropriate time . Over the year many number of software reliability models have introduced. Thus models showing future success approaches to software reliability prediction by controlling, planning the resources during software development process for developing high quality software. It gives confidence about software correctness. Also it minimizes the additional cost of testing and improves software reliability.

(Mallikharjuna Rao & Anuradha, 2016) introduced Identifying and removal of the residual faults are one of the key features in software reliability indexes. (Mallikharjuna Rao & Anuradha, 2016) introduced SRGM is very helpful for software developers and has been widely accepted and applied However, each SRGM contains some undetermined parameters. Which means no single models can get accurate prediction for all cases. So optimization of these parameters is a necessary task. With this motivation, we investigated the improvement and application of ANN with optimization algorithm, namely Genetic algorithm, to optimize these parameters of SRGM. The

performance of our proposed model with optimize parameters compare with the existing models ANN.

The following research questions will be expected answer at the end of this study.

1. How we optimize the parameter of SRGM?
2. How we improve the SRGM using neuro-genetic approach?
3. How our model better predicts the reliability of software than the other SRGM?

1.3 Objectives

The following are the general and specific objective.

1.3.1 General objective

The main objective of this research is improving SRGM by enhancing the parameter of software reliability growth model using Neuro-Genetic based approach.

1.3.2 Specific Objective

To achieve the general objective, the following activity will be performed

- ✓ To study the existing SRGM
- ✓ To identify and define a number of criteria with important level of selection of SRGM.
- ✓ To develop a proposed SRGM using neuro-genetic algorithm.
- ✓ To evaluate the performance of our proposed model using software failure datasets.
- ✓ To compare and contrast the proposed model with the existing SRGM.

1.4 Scope of the study

The aim of this study is to enhancing the parameters of SRGM using neuro-genetic approach to improve the reliability prediction so as to optimize the parameter of SRGM the cause of undetermined parametric problems. In this study an attempt is made to design optimize parameter to find a suitable SRGM with the failure rate. The efficiency of the model concerning decreased failure rate and optimizing the fitness are the major consideration for selecting the appropriate model for reliability growth in propose method

Software reliability is a vast research area then this study is delimited in SRGM that are only focused on certain NHPP SRGM and to evaluate the prediction quality of SRGM are selected from existing literature. Algorithms used in this study is develop using MATLAB.

1.5 Methods

In this thesis work, we follow design science research methodology. Following is the description of each phase:

1.5.1 Problem identification and motivation

In this phase, our research problem is defined and the value of a solution is justified. Various literatures are reviewed to acquire knowledge about the state of the problem and the importance of the solution. Research works that have been done to predict software reliability by using some data sets that will be analyzed and evaluated to get an understanding of the various methods to increase the performance of our model. The gaps in related research works are analyzed and how we fill in the gaps are presented.

1.5.2 Objectives for a solution

The objectives of a solution are inferred from the problem definition or specification. The objectives of our study that are inferred from the problem specification are explained. Various resources have been reviewed to know the state of the problem, the state of current solutions and their efficacy.

1.5.3 Design and development

In this section, the artifact solution is created. This activity is focused on the functionality and design of our models. Feed forward neural network is used for designing the ANN based SRGM. Matlab is used for writing the required source codes.

We collected software failures datasets we are applied for the proposed models. These datasets are found from software projects. This dataset is divided into two parts: training dataset used to train the model and to increase the performance of the system through different parameters; testing dataset to evaluate the system. Two data sets was collected. It will be divided in 70/30 format for training and testing respectively.

Our system consists of four main components: Collection of data sets, Normalization of the data sets, optimization of ANN based SRGM and training parameters using GA. prediction, in turn, encompasses three main phases to predict software reliability. These are:

1.5.4 Demonstration

The developed system is demonstrated by simulating how the developed system to estimate and predict software reliability. The model was implemented with MATLAB package

1.5.5 Evaluation

The developed system is evaluated to measure how well it supports a solution to the problem. To evaluate the system in a rational method, testing datasets were fed into the developed model. Subsequently, the model was evaluated by comparing its output against the observed data using Root Mean Square Error(RMSE).

1.5.6 Communication

In this section, the problems, the artifacts of the designed solution, the effectiveness and other related information are communicated to relevant audiences when appropriate

1.6 Significance of the study

Software reliability is a significant part of software industry; it gives measure to the customer as well as the developer about the faults in the software. The prediction of the reliability of any software is really essential in software industry. SRGM give an estimation to the number of faults that may occur in near future after the delivery of the product and thus the models also provide an induction of when to release the software, it uses the past data gathering in the testing process
The significance of this paper will be the following

- Used to develop a reliable software under a given time and cost constraint

- The software manager also determines the release time of the software with the help of the model.
- Used to measure the quality of the software.
- Used to estimate the duration of the testing time effectively.
- Used to support the project manager to monitor testing
- Helping the researchers to evaluate the reliability of the model

CHAPTER TWO

Literature Review

2.1 Software Reliability Growth Model (SRGM)

As (Inoue & Yamada, 2009*)* introduced a software reliability growth model (abbreviated as SRGM) is fundamental technologies for quantitative software reliability assessment, and playing an important role in software project management for producing a highly-reliable software system. In order to selecting the best SRGM that is compatible with the ANN we should know their predictive power and their mean value function of each model. As (Aggarwal & Gupta, 2014) introduced 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.

Table 2. 1 SRGMs with mean value function

SRGM	Mean value function
Goel–Okumoto model (GO)	$a(1 - e^{-bt})$
Yamada delayed s-shaped model (Y)	$a(1 - (1 + bt)e^{-bt})$
Inflection s-shaped model (I)	a(1-e-bt) / 1+ˇe-bt
Logistic growth curve model (L)	$a/1+be^{-ct}$

As (Saleem, 2013) proposed typically two broad categories of software reliability growth models (SRGMs) include parametric models and nonparametric models. (Singh & Kumar, 2010) introduced most of the parametric models are based on nonhomogeneous Poisson process (NHPP) that has been widely used successfully in practical software reliability engineering. Artificial

Neural Network (ANN) based non-parametric software reliability models estimates the model parameter without any distributions and assumptions.

2.2 Overview of The Artificial Neural Networks

2.2.1 Artificial neural networks (ANNs)

An artificial neural network, or simply neural network, is a type of artificial intelligence (computer system) used to stores information. It generally used to capture the non-linear data of software failure process. ANN based software reliability models for predicting cumulative software number of failures used cumulative software testing time as input and cumulative software number of failures as output.

Feed forward ANN are commonly used architectures in literature which has an input layer, one/more hidden layer and an output layer. In this work, a new ANN architecture is proposed to predict cumulative number of failures in software. The proposed ANN architecture is trained using GA method. Generally, ANNs have the following components.

Neuron: Neuron is the information-processing unit. It is weighted sum of the input signals x_j at the presence of thresholds, passes the sum through the activation function or transfer function of the ANN to process the input signals and generates an output of the neuron of the network. As we see in the figure the neuron is connected layer by layer and connected to each other directly through communication links associated with some weights.

As(Agatonovic-Kustrin & Beresford, 2000) mentioned the artificial neuron is the building component of the ANN designed to simulate the function of the biological neuron. (Tawfiq & Salih, 2014) described the arriving signals, called inputs, multiplied by the connection weights (adjusted) are first summed (combined) and then passed through a transfer function to produce the output for that neuron.

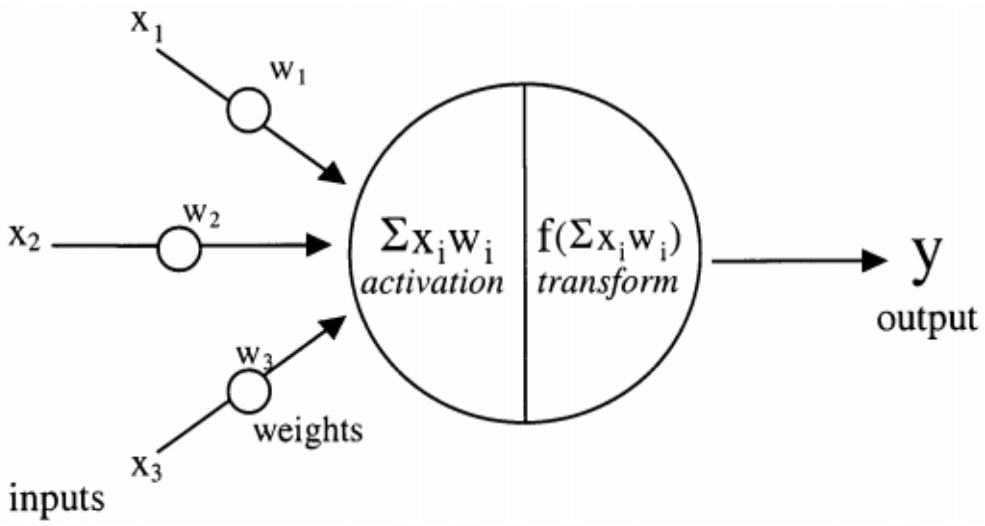


Figure 2. 1 Model of an artificial neuron (Agatonovic-Kustrin & Beresford, 2000)

where net_i is the input signals and w_{ji} is the weight of the connection link between the input xj and neuron of the network.

Network architecture: It determines the network topology applied. The most common ANN topology is forward-feeding connections.

Learning algorithm: ANN learns from training examples by adjust the weight of the network. BPA is a class of supervised learning algorithm which is commonly used. It is a learning techniques are used to update the weights of the network for training the feed forward back propagation network. Recently, GAs have been applied as the learning algorithm for optimization of network weights to reduce errors.

For the general model of artificial neural network, the net input can be calculated as follows

$$y_{in}=x_1.w_1+x_2.w_2+x_3.w_3\ldots x_m.\ w_m$$

$$\text{Net input } y_{in}=\sum m_i x_i.\ w_i y_{in}=\sum_i^m x_i.\ w_i$$

The output of the network calculated by using the activation function over the net of the input.

$$Y=F(y_{in})Y=F(y_{in})$$

Outputs = functions of net input

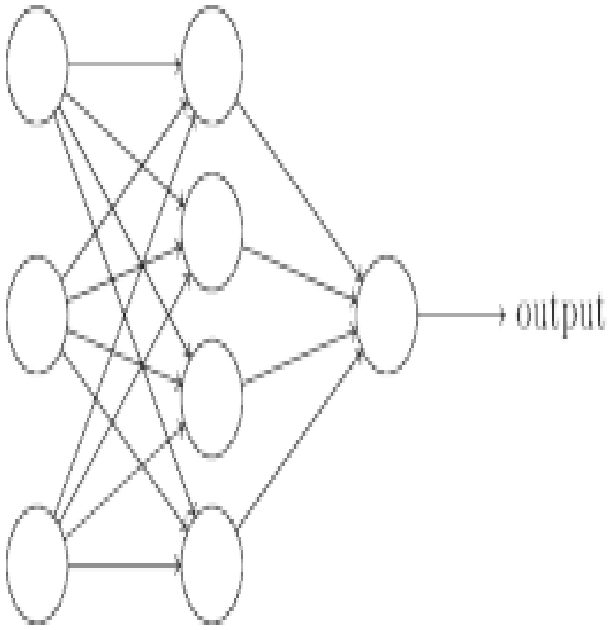


Figure 2. 2 A multi-layer feedforward ANN (Nielsen, 2015)

As we have seen in the above figure the left side the layer in this network is the input layer, and the neurons of the layer are input neurons. The right side or output layer contains the output neurons. The middle layer is a hidden layer. ANN is feedforward if there exists a method which numbers all the nodes in the network such that there is no connection from a node with a large number to a node with a smaller number. Feedforward neural network have only one diction of the network.

ANN is used to an artificial intelligence that is used to processes and stores information. It works by creating connections between mathematical processing elements called neurons, knowledge is encoded into the network through the strength of the connection between different neurons called weight. ANN based models for predicting the reliability of software's. Feed forwarded ANN are commonly used architecture in literature which has input layer, hidden layer and an output layer.

2.3 Genetic Algorithm

As (Rahman & Setu, 2015) introduced Genetic algorithm is robust evolutionary optimization search techniques which follows natural genetics to find global optimal solution. It is a global search method it uses selection, crossover and mutation. It is used for optimization procedures, which is better to find values close to global optimization. Most of the ANN based software reliability models available in literature are trained using back propagation algorithm. It is a gradient based algorithm which suffers from local optimum problem GA technique is widely used to solve the local optimum problem. Genetic algorithms used for optimizing and learn features of biological evolution. An algorithm requires a few base components (Montana & Davis, 1989) The characteristics of Genetic Algorithm are as follows:

- Genetic algorithm operates on parameters of the code.
- The genetic algorithm starts with many points; thus it prevents the search process from converging to local optimal solution effectively.
- It calculates the fitness value by objective function without additional information and relies little on the problem.
- The optimization rules of genetic algorithm are determined by probability, but not deterministic.
- Genetic algorithm operates an efficient heuristic search in the solution space rather than an exhaustive or complete random search.
- There's no limit on genetic algorithm for the function to be optimized. It does not require to be continuous or micro.
- It can both be an explicit function of mathematical expressions and Implicit function like the neural network or mapping matrix.
- The genetic algorithm has the characteristics of parallel operation; thus it can improve the calculation speed through the large scale parallel operation.
- The genetic algorithm has the characteristics of simply compute and strong function and is more suitable for optimization of large scale complex problems.

A. Evaluation

As (Rahman & Setu, 2015) introduced Each member of the population is then evaluated; calculate a ‘fitness’ for that individual and its value is calculated by how well it fits with the desired requirements.

B. Selection

Selection helps by removing the bad designs and having the best individuals in the population crossover by combing the selected individual create new offspring’s. As (Rahman & Setu, 2015) introduced there are a few different selection methods but the basic idea is the same, make it more likely that fitter individuals will be selected for the next generation.

C. Crossover

By combing the selected individual create new offspring’s. As (Rahman & Setu, 2015) introduced The hope is that by combining certain traits from two or more individuals, an even ‘fitter’ offspring will be created which will inherit the best traits from each.

D. Mutation

As (Rahman & Setu, 2015) introduced Add a little bit randomness into the populations’ genetics otherwise every combination of solutions that can create would be in the initial population**.**
Most of the ANN based models available in literature are trained using back propagation algorithm. It is a gradient based algorithm which suffers from local optimum problem GA technique is widely used to solve the local optimum problem. Steps that are used to train ANN with GA Set weights of the network, Define the network and Use GA as the member of the population finally Train the weight until obtain the target output.

2.4 Literature Review

Models that describe the failure phenomenon and consequent enhancement in reliability due to fault removal are termed as SRGM. Many papers are published addressing the problems with the parameter estimation of traditional SRGM is overcome by soft computing techniques such as Artificial Neural Network, Fuzzy systems, Genetic algorithms.

(Karunanithi, Whitley, & Malaiya, 1992a) first presented neural network based software reliability model to predict cumulative number of failures. They apply cumulative execution time of the software as the input of the network. They consider two different training sets like Prediction and Generalization in their own study. They compared their results with some previous models and get better prediction than those models.

(Bisi & Goyal, 2012) Proposed SRP using Neural Network with Encoded Input they presented that the performance of a neural network system can be significantly improved by combining a network. However, only one data set is used as case study to validate and evaluate the reliability prediction

(Lakshmanan & Ramasamy, 2015) proposed the neural network-based combination model with single input neuron in the input layer, single output neuron in the output layer and two neurons in the hidden layer have been done. The result of their proposed model estimation is better than traditional SRGMs in terms of accuracy. However, the model cannot manage well with major changes that are not reflected in training phase.

An ANN based model which used ensembles was presented in (Kapur, YADAVALLI, KHATRI, & BASIRZADEH, 2011) for SRP. The approach was applied on the two software data sets and experimental results showed that ANN ensembles had better predictive capability than single ANN model and some statistical models.

(Ong et al., 2017) proposed a method to invisage the reliability, ranking and selection of SRGM using particle swarm optimization (PSO). The result had shown that PSO for optimizing SRGM parameter has provided more accurate reliability prediction, but there are no standard approaches to select optimal SRGM. Selection of SRGM requires efficient estimation of reliability parameters which helps in determining the quality

In this Section, some related works using ANN for software reliability modeling and prediction are briefly introduced. Artificial neural network based software reliability model was first presented in (Karunanithi, Whitley, & Malaiya, 1992b) to predict cumulative number of failures. Those authors use Jordan neural network and Elman neural network were used to predict cumulative number of failures in software taking testing time as input of the models. The results obtained by the models had better prediction capability than some statistical software reliability models.

An ANN based model to predict next-failure time was presented in (Cai, Cai, Wang, Yu, & Zhang, 2001) Recent 50 failure times were used as input of the model to predict the next-failure time as output. Number of input nodes and hidden nodes were varied to evaluate the prediction accuracy of the model. The result was found that prediction capability of a model depends on the nature of data sets.

A dynamic weighted combinational model (DWCM) based ANN model was presented in (Su & Huang, 2007) predict the reliability of the software. They use activation functions in the hidden layer were used depending upon the software reliability growth models (SRGM). The model had been applied on two data sets and it was found that results were better than some statistical models. An ANN based model which used ensembles was presented in (Zheng, 2009) for software reliability prediction. The approach was applied on two software data sets and experimental results showed that ANN ensembles had better predictive capability than single ANN model and some statistical models.

(Lo, 2009) developed software reliability prediction model using artificial neural network. They examine several models without assuming some unrealistic things. Bayesian regularization is applied to train the network, they commented that their approach produced less average relative prediction error than the other prediction techniques.

Software reliability growth models (SRGM) based on non-homogeneous poison processes (NHPP) using a unified theory (Saleem, 2013) was presented in which incorporates the concept of multiple change-points. The model was applied on three software failure data sets and found better prediction result than some existing SRGM.

(Bhuyan et al., 2016) presented an approach for predicting software reliability. These approach focuses on two types of experimentation; a) next-step prediction or short-term prediction of the reliability and b) the end-point prediction is performed at the end of a future testing and debugging session. The short term predictions result shows better accuracy than end-point predictions for both the data sets.

In literature, a number of software reliability models exist which are used to predict cumulative number of failures in software. All the ANN -based models use back propagation algorithm to train the network. Back propagation is a gradient based technique which suffers from local optima problem. In this paper, Genetic algorithm method is used to train the network which is a gradient based global optimization

CHAPTER THREE

Methodology

3. Introduction

This paper addresses the ANN based software reliability model problems that occurred when using a cumbersome trial-and-error procedure by adopting a methodology based on GA method. Which is used to optimizing the ANN parameters includes, activation function, training algorithm, learning rate, momentum rate and number of epoch as depicted in Figure 3.1.

Each data sets consists of three parts training, test and prediction samples. As (Arifovic & Gencay, 2001) introduced the training sample is utilized during the local minimization stage, while the test sample is used to evaluate a fitness value of a given network. Finally, the prediction sample of a dataset is used only for evaluating network predictive power.

The ANN parameters used the success of the training phase. Back propagation training is a gradient descent algorithm. It tries to improve the performance of neural network by reducing its error along its gradient, but it takes more time to reach the neighborhood of an optimal solution. On the other hand, genetic algorithm used for global search methods. It investigates the entire search space. Hence, they reach faster the region of optimal solution.

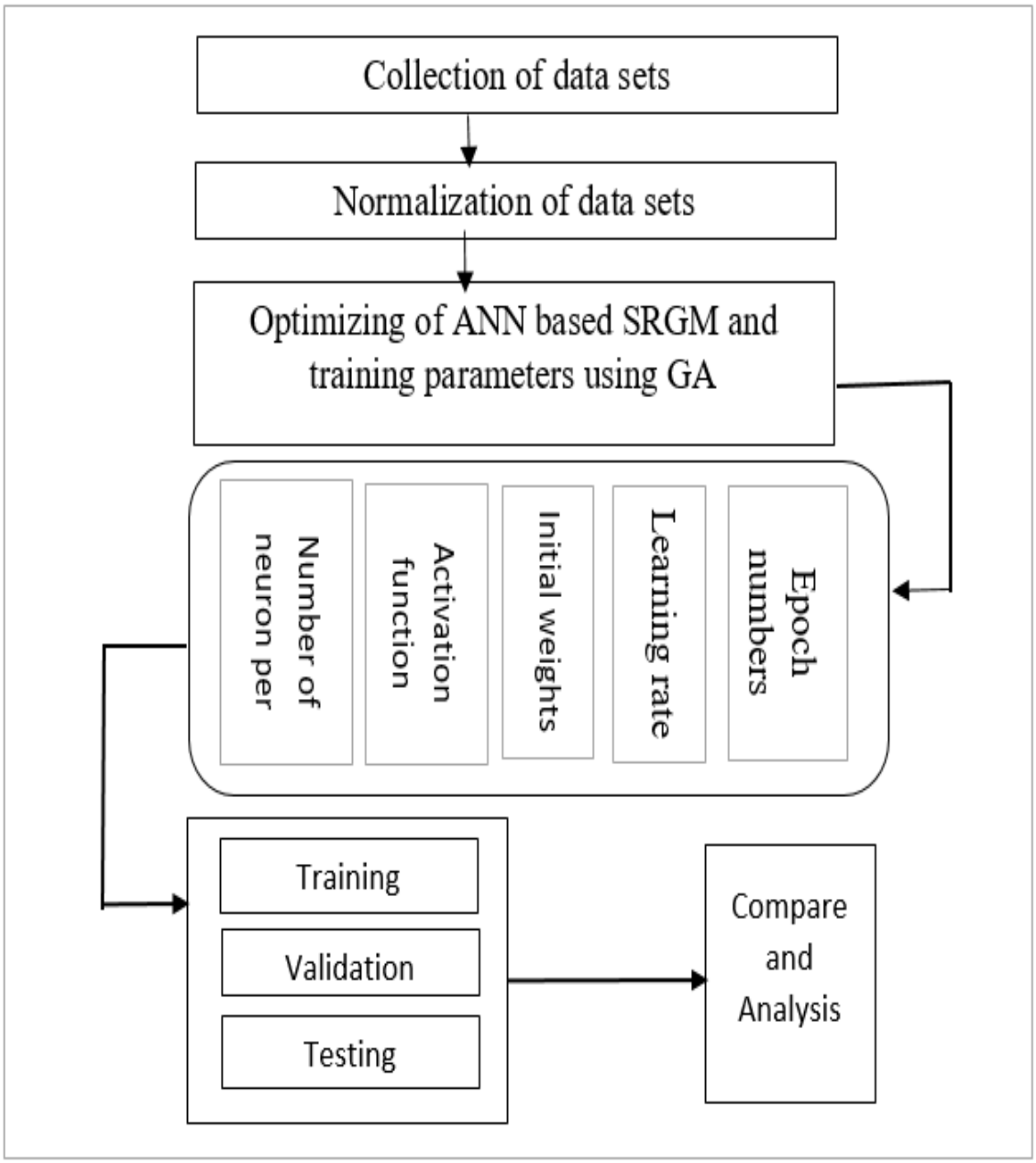


Figure 3. 1 Flowchart of the proposed model

3.1 Software failure data sets

The datasets we will obtained from Handbook of Software Reliability Engineering. The data source obtained from http://www.cse.cuhk.edu.hk/~lyu/book/reliability/data.html

Two data sets DS1 and DS2 are used to check the performance of our proposed model based the proposed GA. DS1 was collected from a real-time command and control application and DS2 was collected from Brazilian Electronic Switching System of Assembly Language [Kano93b, Mart91, Lapr91]. Each data point within a set consists two parts. One is the cumulative execution time and the other is the corresponding cumulative number of failures. Most of the literature is used this data sets.

3.2 Normalized of software failure data sets

The datasets of software failure cannot be supplied to the ANN and we must have normalized these datasets. So we need to normalized data with respect to their maximum values. As (Abdalla, Elfaki, & AlMurtadha, 2014) introduced the training, validation and testing dataset were scaled to the range of (0–1) using the modified MATLAB functions 'premnmx' and 'tramnmx'. The following equation was used for the purpose:

$$x_{ni} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \qquad (1)$$

Where x_i is the real-world input value, x_{ni} is the scaled input value of the real-world input value x_i and x_{min} and x_{max} are the corresponding minimum and maximum values of the unscaled dataset.

As (Abdalla et al., 2014) introduced The network predicted values, which were in the range of (0– 1), were transformed to real-world values using the modified MATLAB function 'postmnmx'. The equation below was used for the purpose:

$$x_{i=}\, x_{ni}\,(x_{max}\text{-}x_{min}) + x_{min} \qquad (2)$$

3.3 Model Architecture

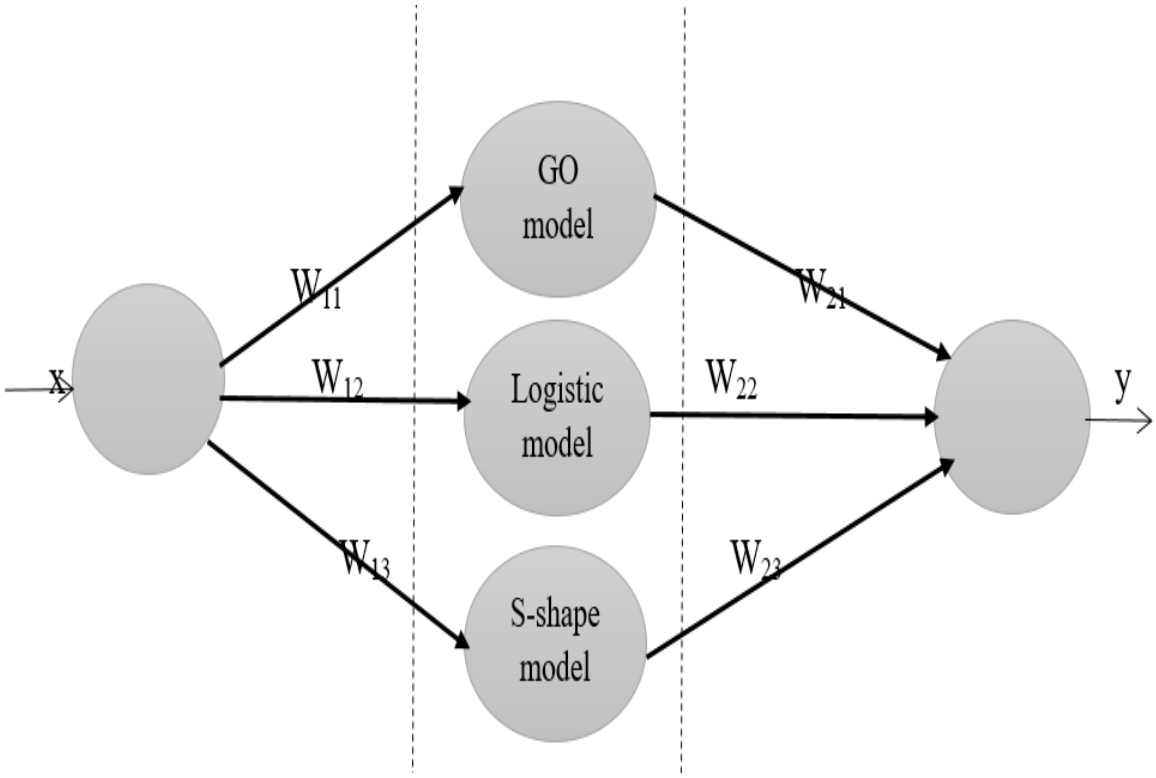


Figure 3. 2 Architecture of the proposed model

Where, w_{11}, w_{12}, w_{13}, w_{21}, w_{22} and w_{23} (>0) are the weights of the proposed ANN-based software reliability model and the weight values are found by genetic training algorithm

Artificial neural networks are generally used to capture the non-linear behavior of software failure process. ANN based models for predicting cumulative number of failures in software used testing time (In proper units) as input and cumulative number of failures as output. As (Bisi & Goyal, 2015) introduced Feed forward ANN are commonly used architectures in literature which has an input layer, one/more hidden layer and an output layer. In this work, a new ANN architecture is proposed to predict cumulative number of failures in software. The proposed ANN architecture is trained using GA method. The structure of feed forward network has neurons arranged in layers and each layer have connections (weights) from the neurons at its previous layer. Fundamentally,

an FNN optimization/learning/training is met by finding of an appropriate network layers (a function) and the weights (the layer of the function).

ANN architecture of consists determination of a number of optimal neurons in hidden layers creating a challenging puzzle that makes us use the trial and error method. (MirRokni, 2017) introduced ANN training is an optimization process to determine the optimal values of weights and biases. Applying Genetic Algorithm gives a better solution for these problems. GA is optimized through random search technique. It reduces chance of convergence to a local optimum.

A simple multi-layer Feed-Forward Neural Network is used to design our proposed model. We use the data goes in a forward direction, starting from the input layer for accepting the input elements to the output layer to produce the output through hidden layer to map the input and output elements. The proposed ANN-based software reliability models with activation function is constructed with single input and output layer each has single neuron and three hidden layers each has single hidden neuron. The structure of the adopted neural network is shown in the above figure

3.4 The neural networks combination model.

(Wang & Li, 2010) proposed the prediction quality of different software reliability models on different failure data sets is not the same, that is to say, there is no an optimal model that can be applicable to all the software failure data sets. From some literatures, we know that a linear combination model has a better software reliability prediction results than those single models, but the weights of the base models in the combination models cannot be optimized automatically. In order to address this problem, we propose to combine the base models with neural networks, since neural networks have powerful fitting ability and generalization.

As (Wang & Li, 2010) introduced the software reliability models with good prediction results in engineering practice will be chosen as the base models in the neural networks, and appropriate activation functions will be selected for each base model. Genetic algorithm in neural networks can be used to optimize the weights of the base models to improve the software reliability prediction. ANN can learn from training example by using weights. The weights of our model are adjusted by using the proposed GA based on training data in order to minimize the errors of the actual output as compared to the predicted outputs. Most of the ANN based models available in literature are trained using back propagation algorithm. It is a gradient based algorithm which

suffers from local optimum problem. Genetic Algorithm technique is widely used to solve the local optimum problem, in this work, there are three novel aspects. First, a new ANN based software reliability architecture is proposed to predict cumulative number of failures in software taking testing time as input of the model. The purpose is to evenly distribute the inputs of ANN in a specified range for better training of ANN. Second, the proposed ANN based model is trained using Genetic algorithm to predict cumulative number of failures in software. Third, the proposed model is validated using two data sets available in literature. Results of the proposed model are compared with the existing ANN based models available in literature.

3.5 ANN Approaches for Software Reliability Modeling

3.5.1 The selection of the base models

As we mentioned in chapter 2 there are many software reliability models, among the many software reliability models, we choose GO model, logistic curve model, S-Shape model because their performance in software reliability evaluation. The mean value function of These three model are as Eq. (1), (2), and (3).

We develop ANN-based software reliability model with one input neuron in the input layers, one output neuron in the output layers and three neurons in the hidden layers. As (Lakshmanan & Ramasamy, 2015) proposed the number of neurons in the hidden layer is determined by the number of base models selected to construct the neural network combination model.

As (Wu, Han, He, & Wu, 2012) proposed several approaches have been developed to combine various existing software reliability models to produce a dynamic weighted combination model whose prediction accuracy is much better than the component models. We develop the proposed model by using activation function for hidden layer neurons of our model. We use software Cumulative execution time as the input and predicted cumulative failures number of software as the output of our model. We propose GA based to train our model based software reliability model using failure data sets of the software by global optimize the weights and parameters of the ANNs.

Table 3. 1 The mean values of function of the selected models is the following.

$m(t)= a (1-e^{-bt})$	GO model …… ……. (1)
$m(t) = a / 1+ke^{-bt}$	logistic curve model ……. (2)
$m(t)= a(1-(1+bt)e^{-bt}$	S-Shape model …… .(3)

According to the need for the activation functions in our model, the activation functions are designed as simple, continuous, easily differentiable and the function of the output can be mapped approximately as a compound form g(f(x)). The output of our model is based SRGM can be derived from its mean value function as a compound form as the following: Equation, which is the mean value function of the proposed model.

The cumulative execution time (x) is the input of the proposed model and output for our model is the predicted cumulative failure number of the software (y). We use $1 - e^{-x}$, $1 - (1 + x) e^{-x}$ and $1/ (1 + b_1e^{-x})$ as the activation functions for the Three hidden layer neurons of the proposed model. The mean value function of the selected SRGMs is used for developed the activation function in the hidden layer of the neurons.

Linear activation function g(x) = x is used in the output layer neuron of the ANN. Hence, the output of the proposed model can be evaluated as follows.

Design the activation function for each base model. The derivation of the neural network into the software reliability modeling has the following

if we use $f(x)= 1 - e^{-x}$ as the activation function of in the hidden layer and linear activation function g(x)=x in the output layer then we can get the mean value function of G-O model. $Y(t)=W_{21}(1- e^{-W11t})$.

Again, if we use $f(x)= 1/ (1 + b_1e^{-x})$ as the activation function of in the hidden layer and linear activation function g(x)=x in the output layer then we can get the mean value function of Logistic curve model. $Y(t)= W_{22} / (1 + b_1e^{-W12t})$.

Finally, if we use $f(x)= 1 - (1 + x)\, e^{-x}$ as the activation function of in the hidden layer and linear activation function g(x)=x in the output layer then we can get the mean value function of S-shaped model. $Y(t)=W_{23}(1-(1+W_{13}t)\, e^{-W13t})$.

Linear activation function g(x) = x is used in the output layer neuron of the ANN based model. Hence, the output of the proposed model can be evaluated as follows.

$$Y'(t)= W_{21}(1- e^{-W11t}) + W_{22}/ (1 + b_1 e^{-W12t}) + W_{23}(1-(1+W_{13}t)\, e^{-W13t})). \qquad (4)$$

where W_j (> 0) are weights of the proposed model and their values are determined by the GA. Here, $b1$ ($b1 > 0$) are activation function parameters whose values are also evaluated through the learning of the proposed model.

The values of all the weights and parameter of the feed-forward neural network are determined by the GA and the three SRGMs are combined based on these dynamically evaluated weights in accordance with the failure data used to train the feed-forward neural network.

3.6 Training neural network: Back propagation vs. Genetic Algorithms

Training of the networks means finding the optimum or best values of weights and biases of the network. The back-propagation neural network is a supervised learning method that uses a gradient descent method to minimize the error between the predicted output and the target output.

Back propagation is only a local optimum algorithm. Neural network based software reliability growth model on back-propagation (BP) algorithm is a widely used prediction model. In this paper, to improve the prediction accuracy, we optimize the neural network from two aspects. Firstly, according to the mean square error of each iteration of network training, the nodes number of the hidden layer is selected adaptively, which can minimize the mean square error. Secondly, using mean square error function to define the fitness function, improved genetic algorithm (GA) is proposed to train the learning rate and momentum factor dynamically, which includes multi-point crossover and single point mutation. GA instead of backpropagation to find the network weights for a fixed set of connections. Genetic algorithm ensures the best weights to be used to train the network and make them converge faster and reduce the error between desired output and the actual output and hence increases the efficiency of the network.

The overall optimization process of ANN model with GA is shown in figure

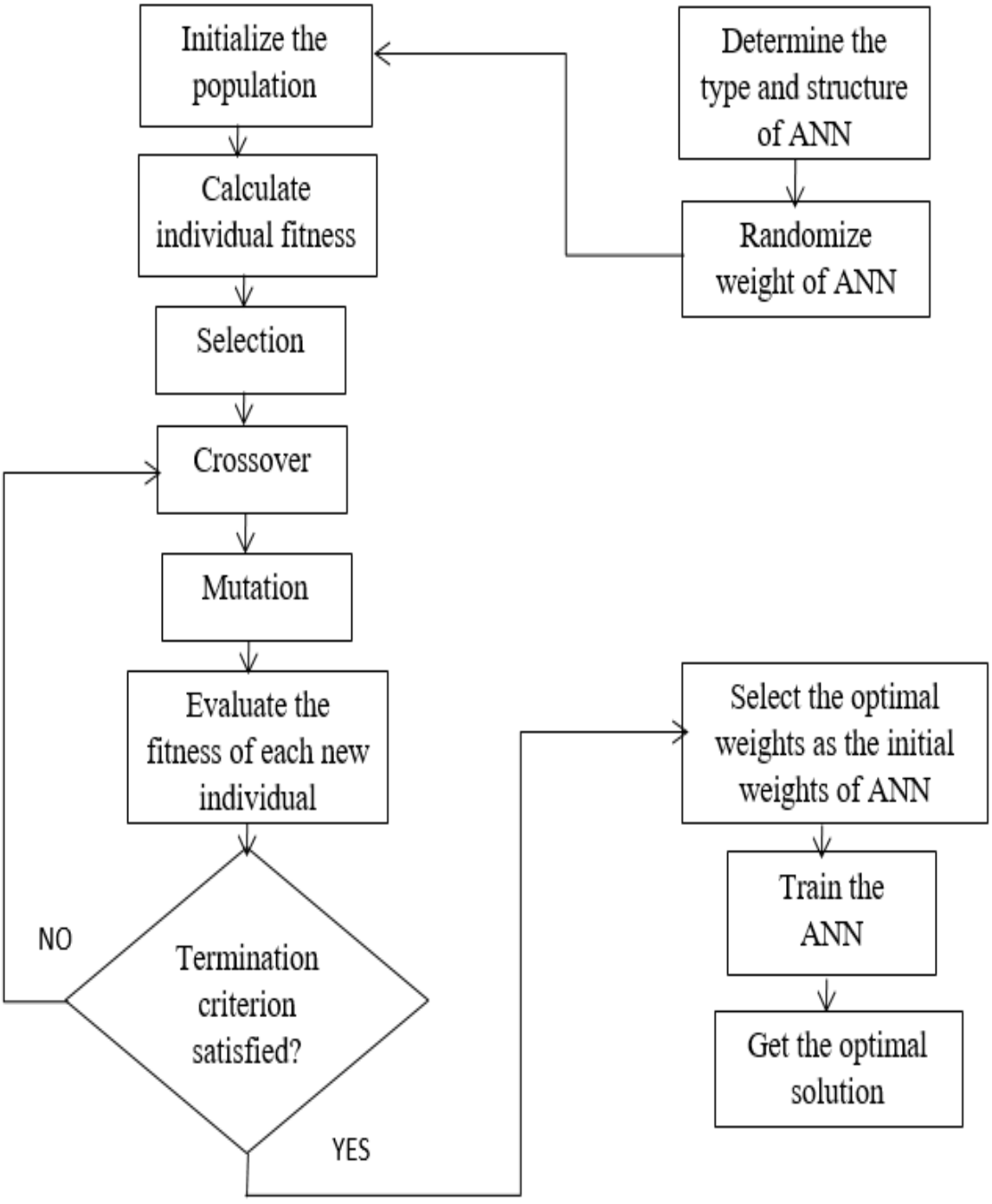


Figure 3. 3 Neural network optimization by the genetic algorithm

As we see in the above figure the first step is determining the type and structure of the network after determining the network the next step is randomize weights of the ANN, from here initialize the population and calculate individual fitness values using fitness function after that we are going to select based on the fitness value which has the highest value, then based on the selection apply crossover and mutation then evaluate the fitness value of each new individual after calculating the fitness value there are two options, if the termination criteria is not satisfied back to the crossover operator, Otherwise the termination criteria is satisfied if it is go to select the optimal weights as the initial weights of ANN then based on the optimal weights train our model finally we can get the optimal solution. it was designed and implemented with MATLAB package.

CHAPTER FOUR

Result and Discussion

4.1 Implementation of the Proposed ANN based SRGM to predict Software reliability

The steps for implementation of the experiment that are needed to design for our models are as follows.

- Select simple software reliability growth model that support models to design the neural network architecture with suitable activation functions for the hidden and output layer neurons.
- Train the network architecture by giving the normalized practical failure data set to the neural network based model using genetic algorithm training algorithm.
- Using the trained neural network, estimate the respective weights of neurons and measure the performance analysis of the proposed ANN-based SRGM.

4.2 GA implementation for training of ANN based model for software reliability prediction.

Step 1: Initialize the parameters of the GA :

Step 2: Set **gn** = 1, where **gn** denotes the current generation number.

Step 3: Encoding the weight and parameter of the ANN into chromosomes.

Step 4: Generate the initial population by initializing the chromosomes for the population.

Step 5: Evaluate the fitness value of each chromosome in the population by considering the fitness function.

Step 6: Selecting the parents from the population by tournament selection process.

Step 7: On the selected parents apply crossover and mutation operations in order to produce offspring's with higher fitness value.

Step 8: If the termination criteria is satisfied, go to step 11. i.e. if gn = max_gen.

Step 9: Increasing **gn** by unity.

Step 10: Go to step 5.

Step 11: Evaluating the fitness value for each of offspring from the population.

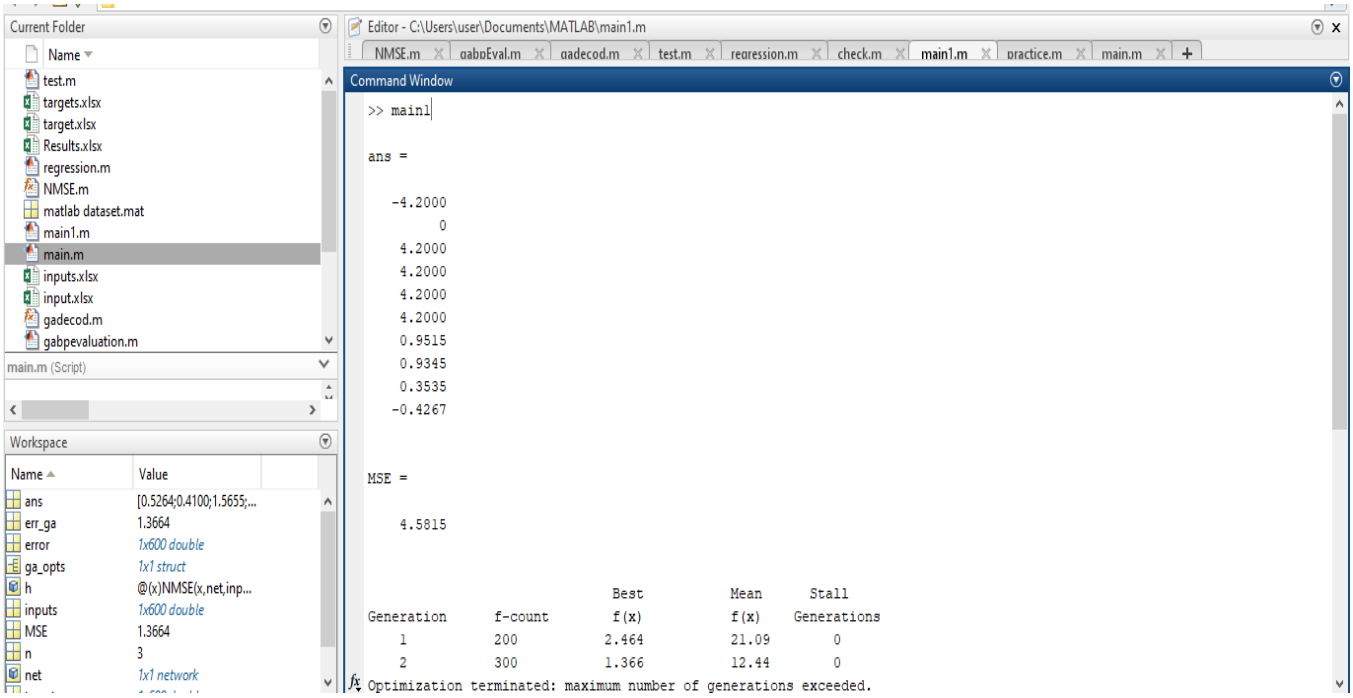
Step 12: Return the chromosome with best fitness value and calculate the error of GA optimized NN (which is the optimal setting of the weights and parameters for the ANN).

Step 13: Stop.

We applied the above implementation for a single input with 3 hidden neurons and a single output in matlab.

4.3 Different Performance Measures

The fitting performance of a software reliability model demonstrates how much fit the model to the software failure data. The performance measure of our model using training of parameters of the SRGM are estimated using a part of the (training of our data) is the first action. Than train our model global optimize the weight of the network using proposed GA. The estimated cumulative failures **yi** at the execution time **ti** is compared with the actual value of cumulative failures **yi** from the dataset. The results of our proposed model has indicated that the new methodology can optimize and training parameters precisely, and resulting in ANN based software reliability model where satisfactory performance in Genetic Algorithm. The fitting performance of our model is measured in terms of Mean Squared Error(MSE) and Root Mean Squared Error (RMSE) as follows



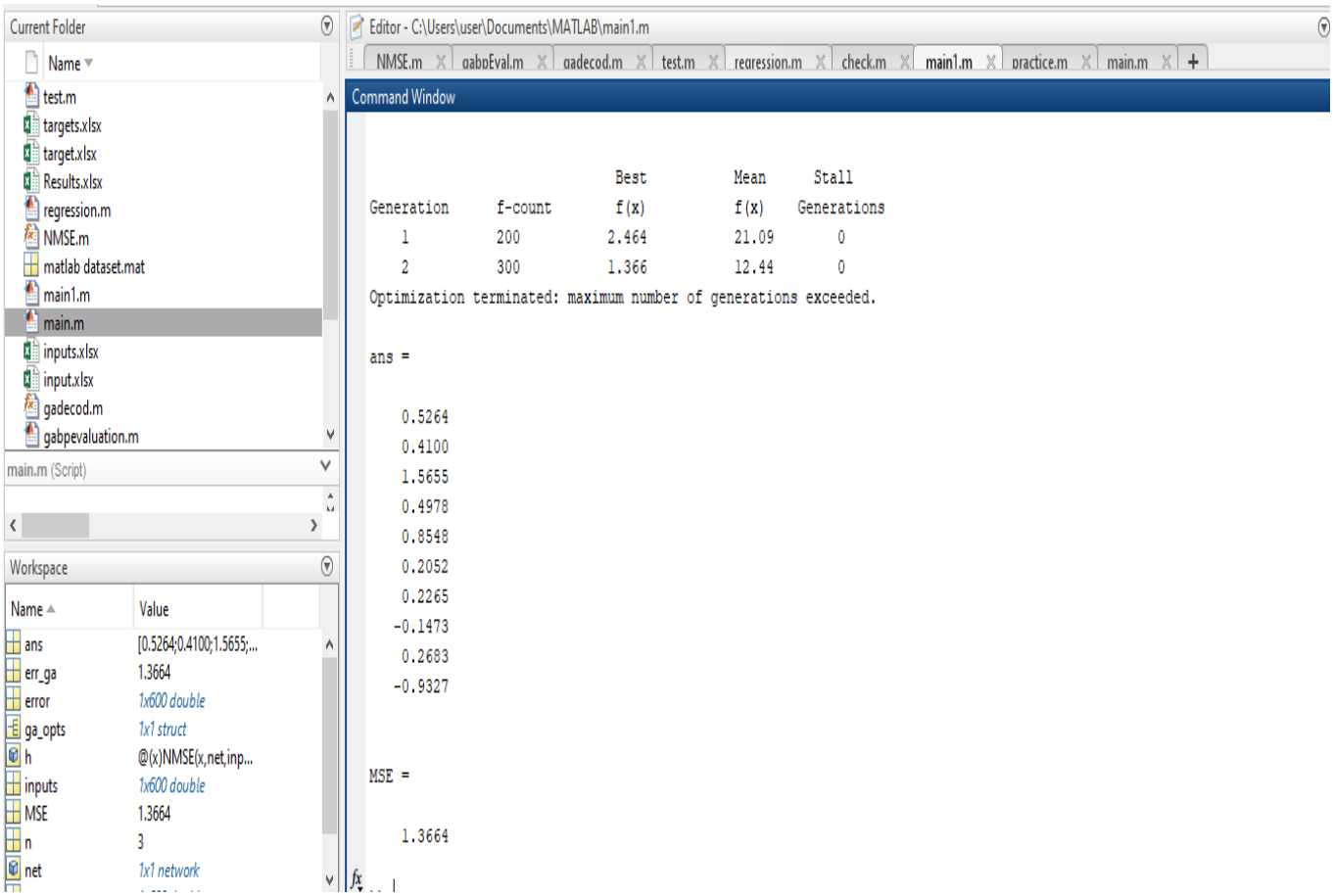
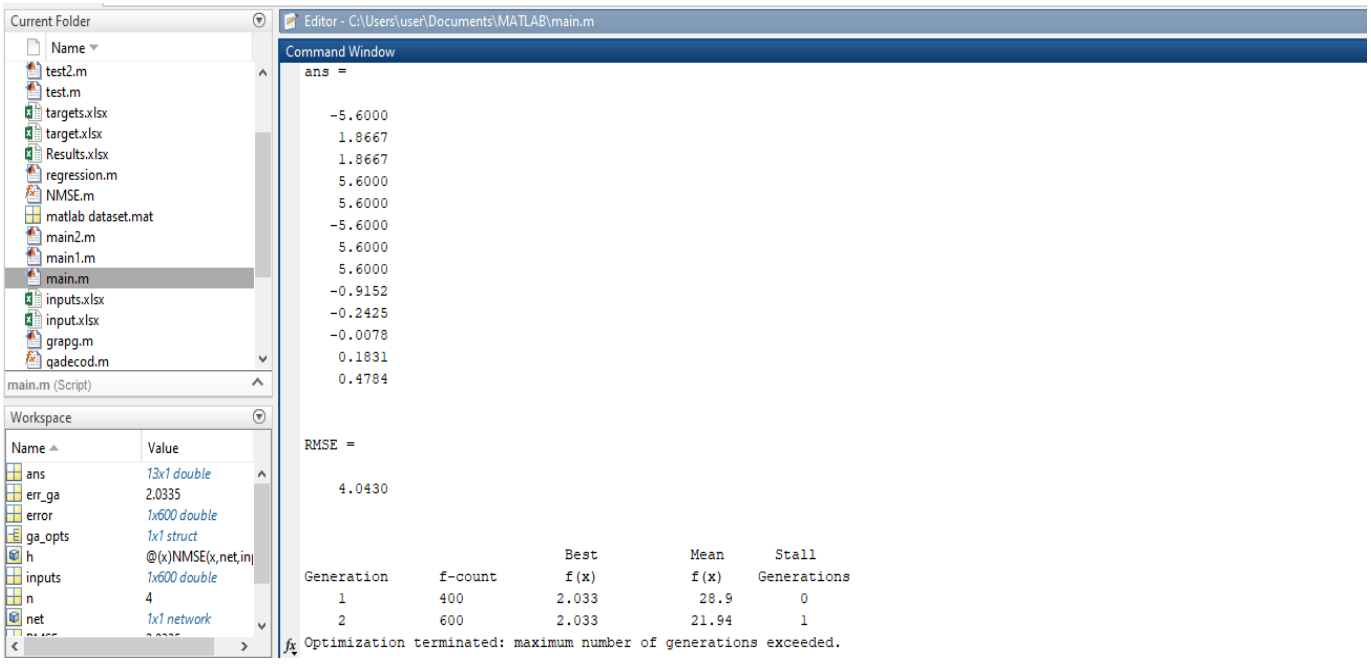


Figure 4. 1 The snapshot of the performance measure values using MSE for Dataset1



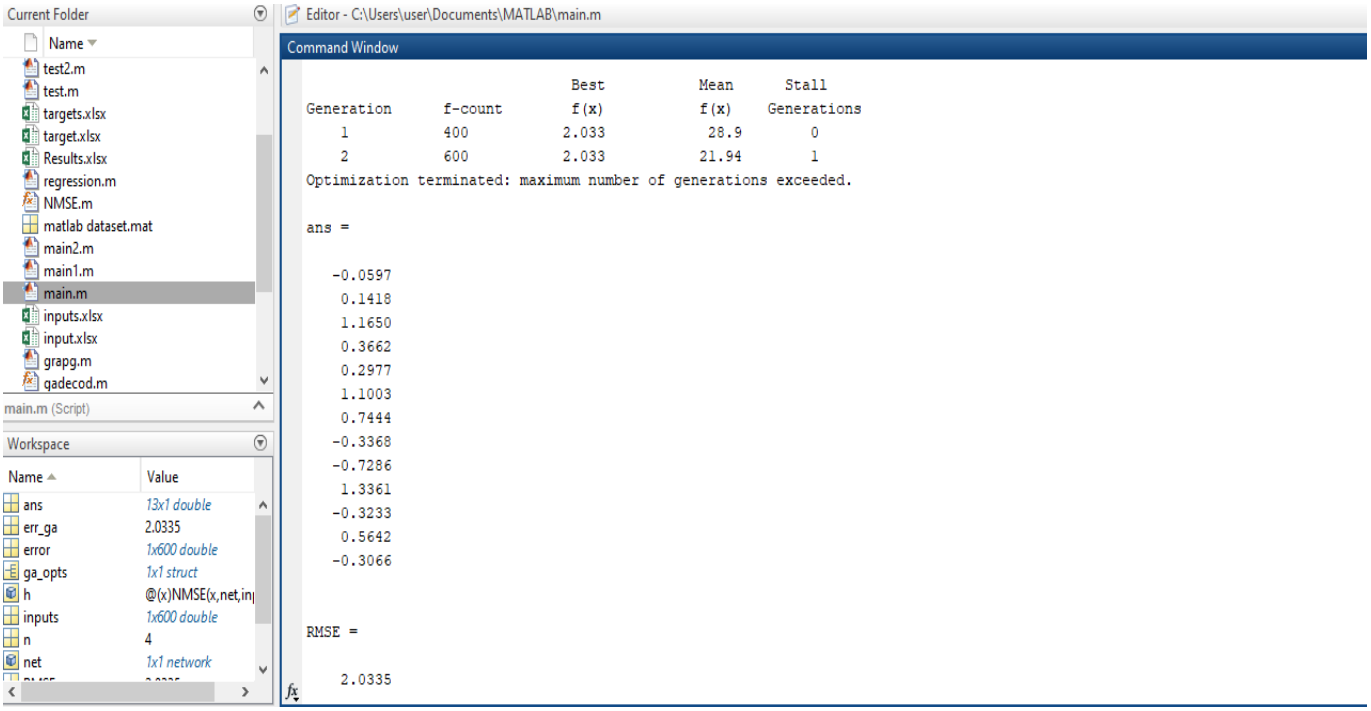


Figure 4. 2 The snapshot of the performance measure values using RMSE for Dataset1

As we see in the above figure the neural network based model performs better in terms of less error in prediction as compared to the normal algorithm and hence it is a better alternative to do software reliability test using genetic algorithm. It can be seen from the figures that the NN method proposed in this paper using genetic algorithm provides a good fit than the normal algorithm.

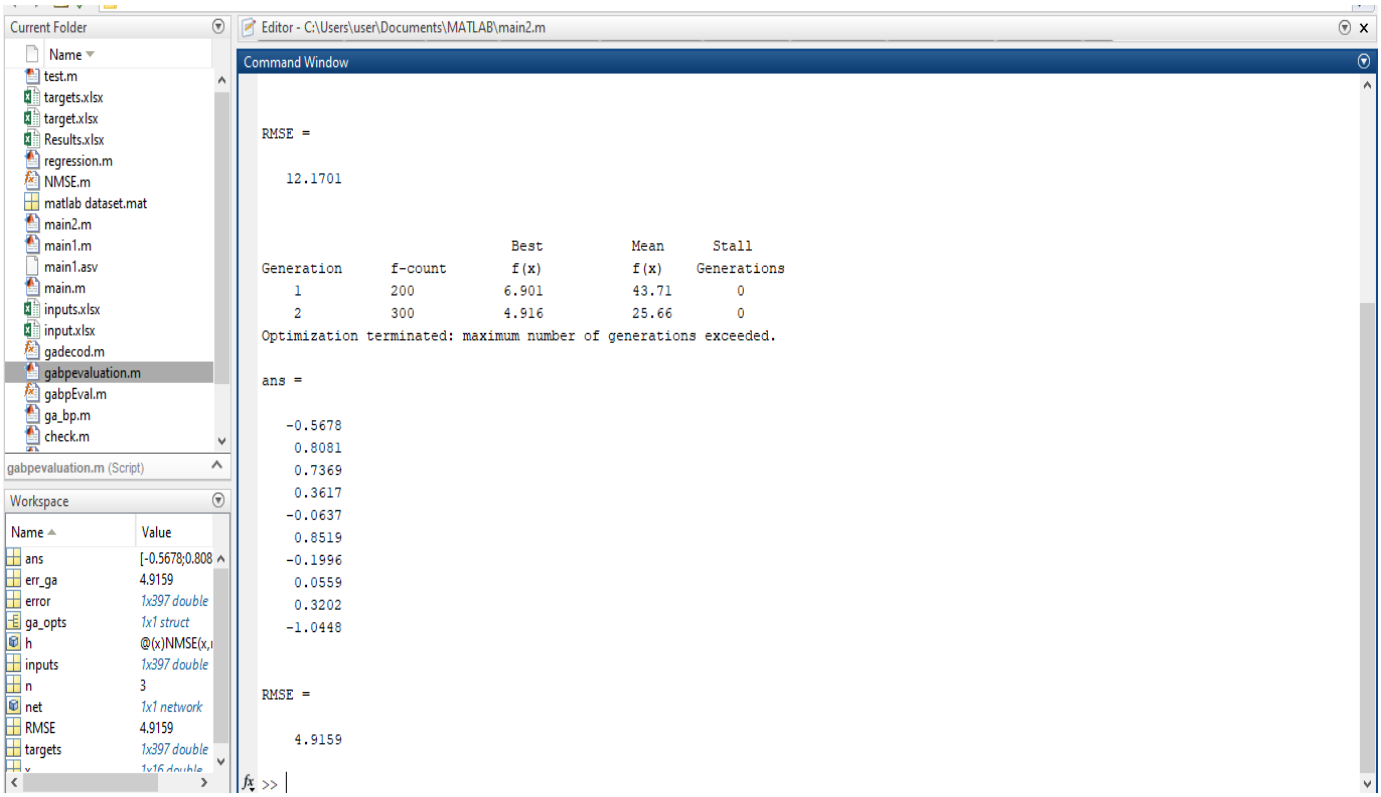
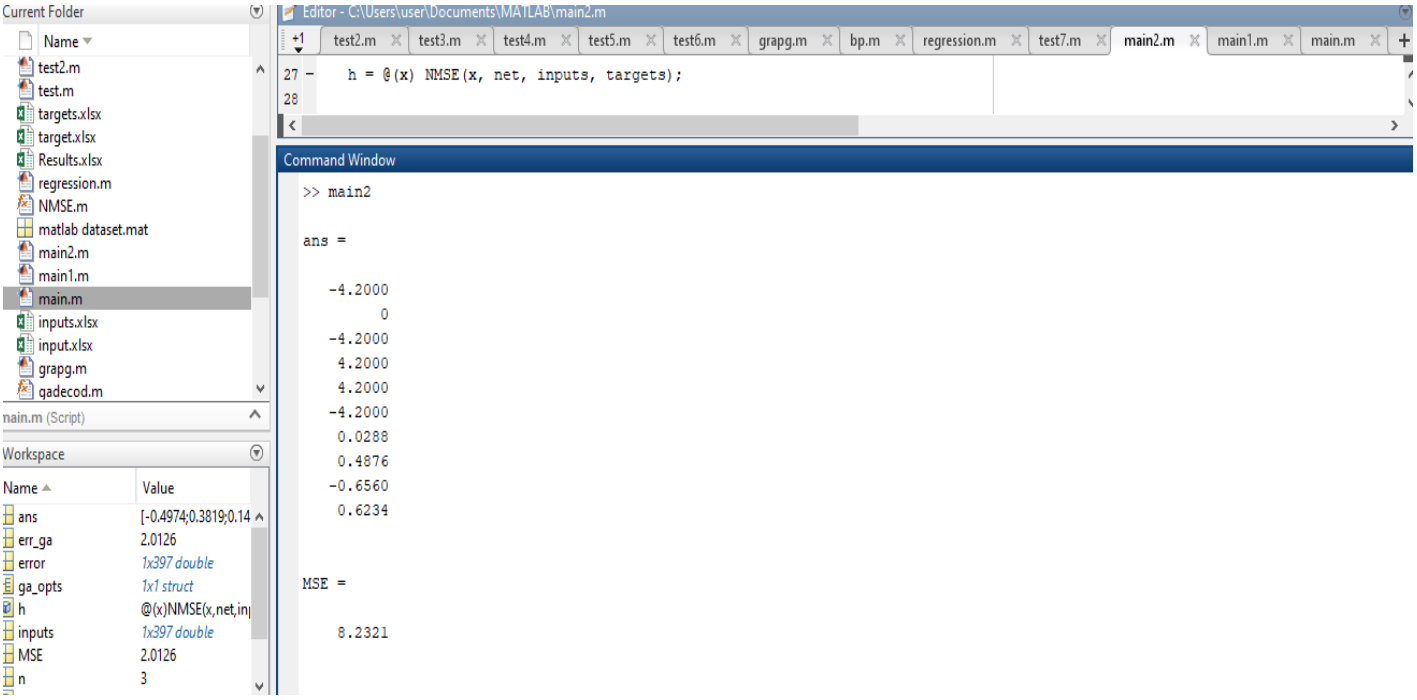


Figure 4. 3 The snapshot of the performance measure values using RMSE for Dataset2



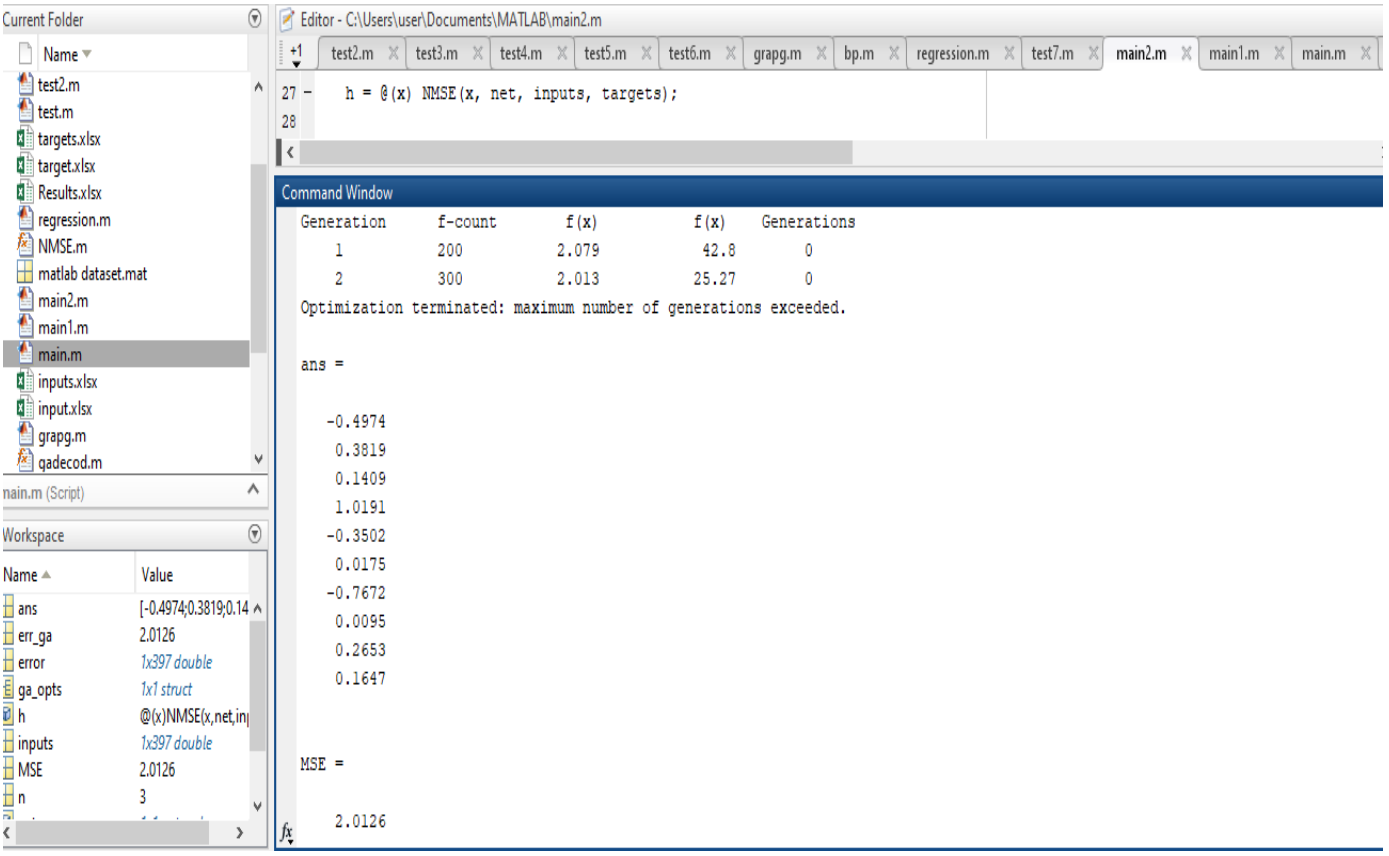


Figure 4. 4 The snapshot of the performance measure values using MSE for Dataset2

As we see in the above figure the neural network based model performs better in terms of less error in prediction as compared to the normal algorithm and hence it is a better alternative to do software reliability test using genetic algorithm. It can be seen from the figures that the NN method proposed in this paper using genetic algorithm provides a good fit than the normal algorithm.

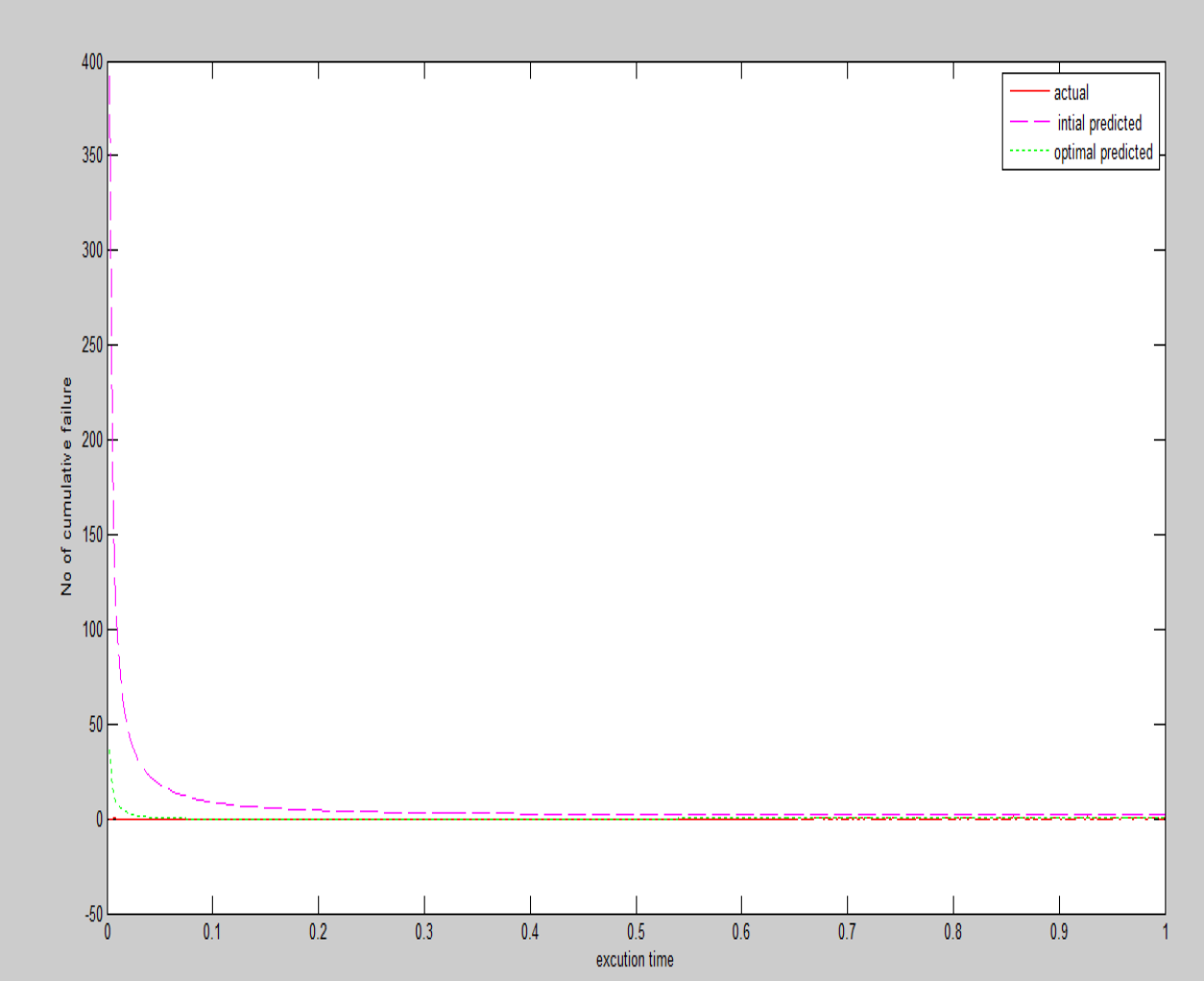


Figure 4. 5 Comparison between Actual data and predicted data in DS1

The graph plots between cumulative execution time and number of cumulative failure. The red color represents actual data; mineral red color represents the initial predicted and green color represents optimal predicted for our proposed model.

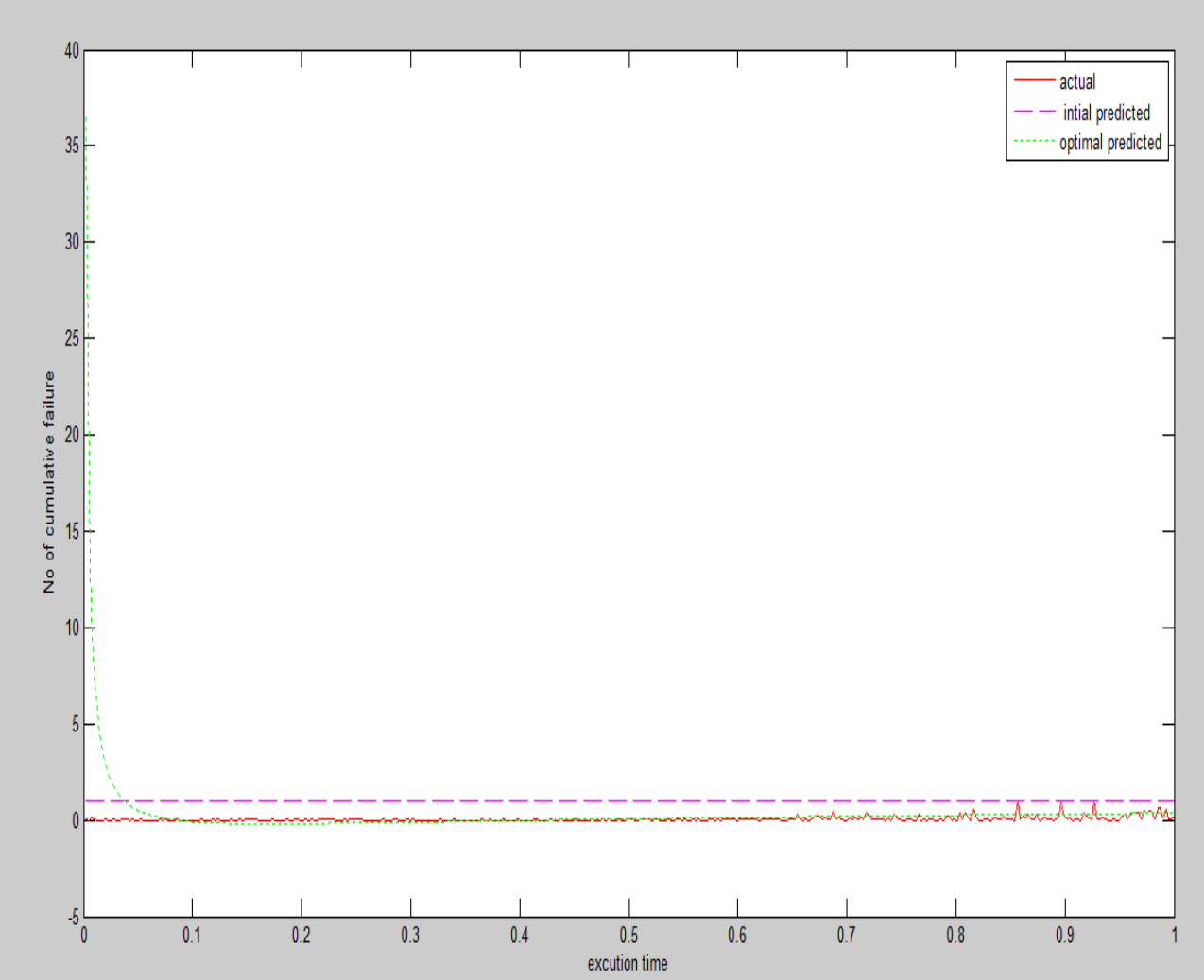


Figure 4. 6 Comparison between Actual data and predicted data in DS2

In the above figures the graphs are the plot between cumulative execution time and number of cumulative failure for both datasets. The red color represents actual data; mineral red color represents the initial predicted and green color represents optimal predicted for our proposed model. As we see in the diagram the predicted accuracy is increased in optimal solutions which means predicted accuracy is increased by using optimization algorithm.

In both data sets it can be seen from the figures that the proposed model using genetic algorithm provides good fit than back propagation algorithm. results are shown in the following tables

Data sets	Numbers of input neuron	Numbers of input neuron	Numbers of hidden neuron	MSE	RMSE
DS1	1	1	3	4.5815	4.0430
DS2	1	1	3	8.2321	12.1701

Table 4. 1 Results for Backpropagation Algorithm

Data sets	Numbers of input neuron	Numbers of input neuron	Numbers of hidden neuron	MSE	RMSE
DS1	1	1	3	1.3664	2.0335
DS2	1	1	3	2.0126	4.9159

Table 4. 2 Results for Genetic Algorithm

The above tables show that the performance of our models under comparison in terms of MSE and RMSE using DS1 and DS2. Table 1 shows the result of back propagation learning algorithm and table 2 shows that the result of genetic algorithm. The smaller value of measure performance indicated better accuracy. As we see in the above result the proposed model has smaller values than the other approaches in both datasets. So better software reliability prediction can be achieved if we train the proposed ANN based software reliability model using GA. Hence, from table 4.1 and table 4.2, shows that the proposed model in GA has better fitting and predictive accuracy than the model in BPA.

	BPA	GA
MSE	4.5815	1.3664
RMSE	4.0430	2.0335

Table 4.3 comparison of Backpropagation algorithm and genetic algorithm in DS1

	BPA	GA
MSE	8.2321	2.0126
RMSE	12.1701	4.9159

Table 4.4 comparison of Backpropagation algorithm and genetic algorithm in DS2

The ability to set GA in the train function is not currently available in neural network toolbox. It tolerance for minimum change in fitness function before terminating algorithm to 1e-8 and displaying each iteration's results. And also it has its own function.

This function may return the mean squared error and root mean squared error on the outputs and the targets as ga requires a function handle that only return a scalar value.

1. All of the weights are randomly changed
2. The resulting error is calculating
3. If the new error is not lower than the existing error, the new set of weights is discarded the algorithm goes back to step1
4. If the new error is lower than the existing error, the new set of weights are accepted.

In the above tables and figures, our model is shown. the error rate is determining in terms of MSE and RMSE during training. It shows how the error rate gradually decrease in genetic algorithm as compared to backpropagation algorithm.

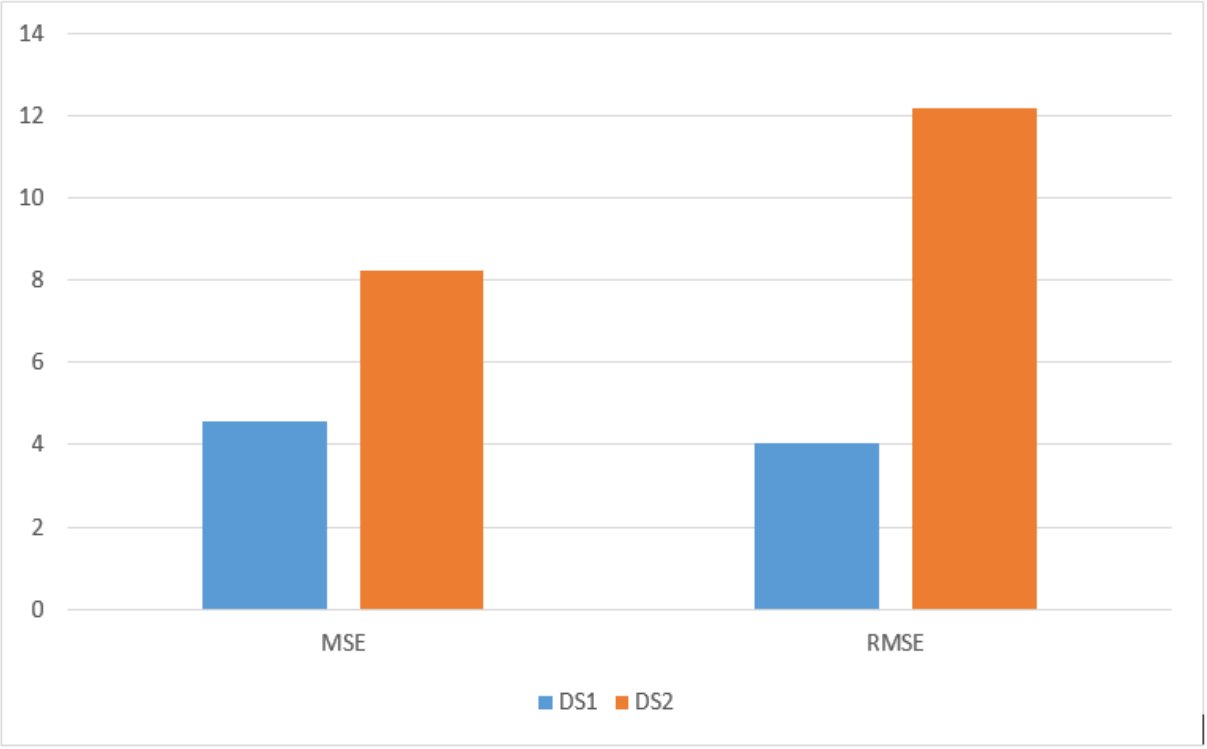


Figure 4. 7 Results that shows the BPA in both datasets

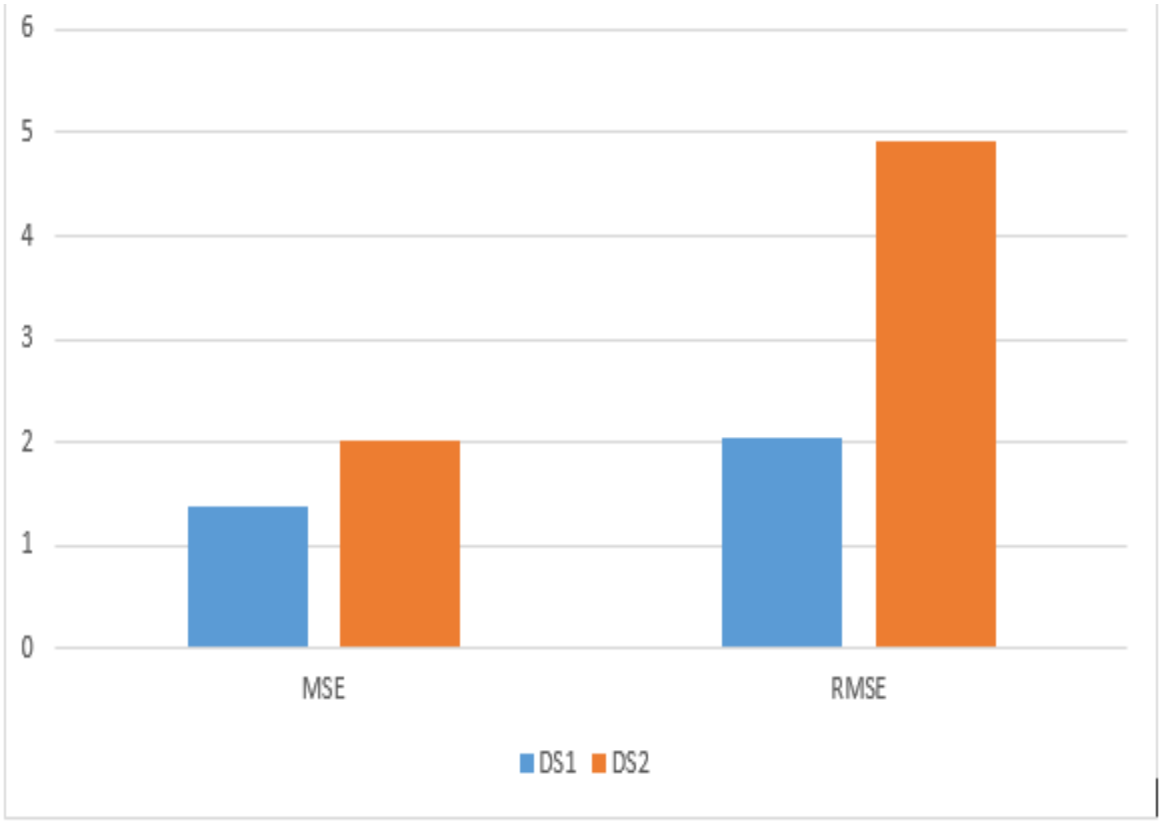


Figure 4. 8 Results that shows GA in both datasets

As we see from the above tables and figures, we find that the excellent prediction ability compared to the observed values from the datasets. All MSE and RMSE values of DS1 are smallest among the values of DS2. It has the best software reliability prediction ability because of its minimum MSE and RMSE values as compared to the values of DS2. So the prediction power of the model is increased wit increase the amount of datasets.

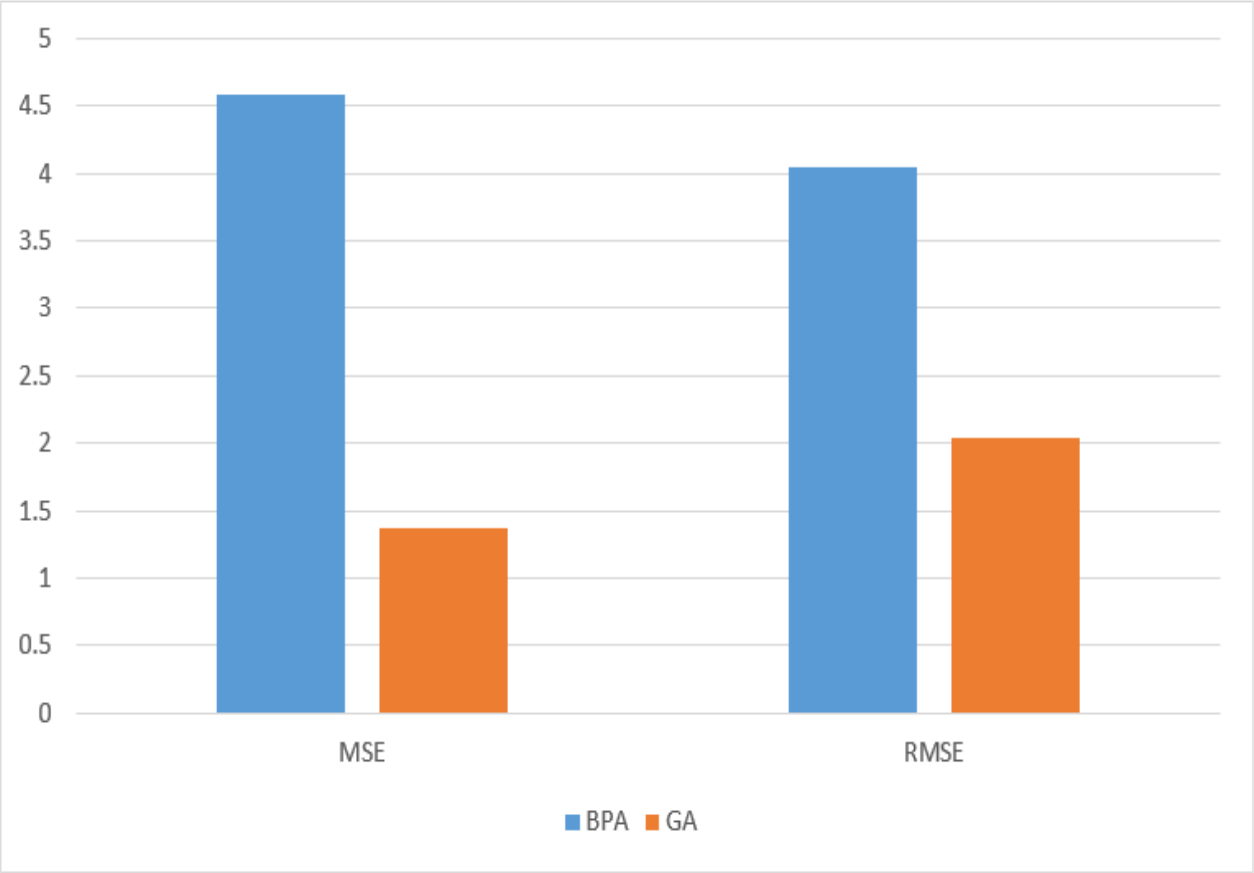


Figure 4. 9 Comparison of backpropagation algorithm and genetic algorithm in DS1

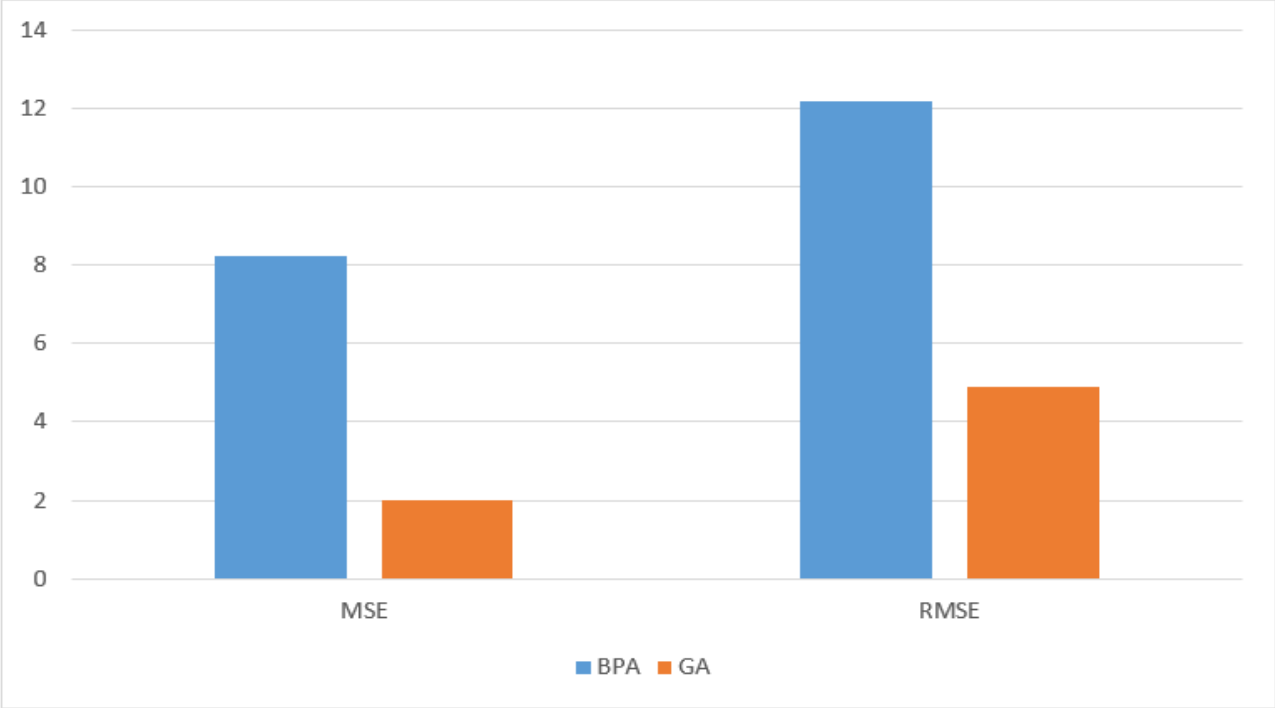


Figure 4. 10 Comparison of backpropagation algorithm and genetic algorithm in DS2

As we see in the figures, it has the best software reliability prediction ability because of its minimum MSE and RMSE value is compared to the other software reliability models. Hence, it is also proved for DS1 that the proposed model has the best software reliability prediction power than the other software reliability models. The predictive power of the model is increased with increase the amount of datasets.

CHAPTER FIVE

Conclusions and Recommendations

5.1 Conclusions

In this paper, ANN based software reliability estimation and prediction is proposed. We used neuro-genetic approach for ANN based software reliability model global optimize the weight of network using proposed GA. We use our proposed GA optimized trained the model in order to predict the reliability of the software. first, we train our model using BPA and predict the software reliability. Then, we use the proposed GA to train our model global optimize the weight of the network. We present the comparison between the two learning algorithms when they are applied to train the proposed model.

In this study we presented the result through two real software failure data sets. Experimental studies show that our proposed model gives better software reliability prediction than the other ANN based software reliability models. The observations conclude that our model performs better in terms of less error in prediction as compared to existing models and hence it is a better alternative to do software reliability test using neuro-genetic approaches. Moreover, the proposed model significantly gives better fitting and prediction accuracy if we use the proposed GA to train the ANN based software reliability prediction model, we get better software reliability predictions from the same ANN based software reliability model. ANN based software reliability model gives better result for larger datasets than smaller datasets.

Our models are easily compatible with different smooth trend data set and projects. We have implemented the program in MATLAB. But the programs can be implemented in other languages such as Java, Python etc.

5.2 Recommendations

Software reliability can be predicted using neuro-genetic system. In addition to neural network model advance machine learning techniques can be applied further. Further, research can be extended by developing model to predict software reliability by introducing advanced machine learning techniques applied to a large category of failure datasets of real life industrial projects. And also plan to focus on the cost benefit analysis of the model that will help to determine whether a given software reliability prediction model would be economically viable in realistic environment.

Reference

Abdalla, Osman Ahmed, Elfaki, Abdelrahman Osman, & AlMurtadha, Yahya Mohammed. (2014). Optimizing the multilayer feed-forward artificial neural networks architecture and training parameters using genetic algorithm. *International Journal of Computer Applications, 96*(10), 42-48.

Agatonovic-Kustrin, S, & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of pharmaceutical and biomedical analysis, 22*(5), 717-727.

Aggarwal, Gaurav, & Gupta, VK. (2014). Software reliability growth model. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(1).

Andersson, Carina. (2007). A replicated empirical study of a selection method for software reliability growth models. *Empirical Software Engineering, 12*(2), 161.

Anjum, Mohd, Haque, Md Asraful, & Ahmad, Nesar. (2013). Analysis and ranking of software reliability models based on weighted criteria value. *IJ Information Technology and Computer Science, 2*, 1-14.

Arifovic, Jasmina, & Gencay, Ramazan. (2001). Using genetic algorithms to select architecture of a feedforward artificial neural network. *Physica A: Statistical mechanics and its applications, 289*(3-4), 574-594.

Bhuyan, Manmath Kumar, Mohapatra, Durga Prasad, & Sethi, Srinivas. (2014). A survey of computational intelligence approaches for software reliability prediction. *ACM SIGSOFT Software Engineering Notes, 39*(2), 1-10.

Bhuyan, Manmath Kumar, Mohapatra, Durga Prasad, & Sethi, Srinivas. (2016). Software Reliability Assessment using Neural Networks of Computational Intelligence Based on Software Failure Data. *Baltic Journal of Modern Computing, 4*(4), 1016.

Bisi, Manjubala, & Goyal, Neeraj Kumar. (2012). Software reliability prediction using neural network with encoded input. *International Journal of Computer Applications, 47*(22), 46-52.

Bisi, Manjubala, & Goyal, Neeraj Kumar. (2015). *Predicting cumulative number of failures in software using an ANN-PSO based approach.* Paper presented at the 2015 International Conference on Computational Intelligence and Networks.

Cai, Kai-Yuan, Cai, Lin, Wang, Wei-Dong, Yu, Zhou-Yi, & Zhang, David. (2001). On the neural network approach in software reliability modeling. *Journal of Systems and Software, 58*(1), 47-62.

Inoue, Shinji, & Yamada, Shigeru. (2009). *Two-dimensional software reliability measurement technologies.* Paper presented at the 2009 IEEE International Conference on Industrial Engineering and Engineering Management.

Kapur, Parmod Kumar, YADAVALLI, VS SARMA, KHATRI, SUNIL KUMAR, & BASIRZADEH, MASHAALLAH. (2011). Enhancing software reliability of a complex software system architecture using artificial neural-networks ensemble. *International Journal of Reliability, Quality and Safety Engineering, 18*(03), 271-284.

Karunanithi, Nachimuthu, Whitley, Darrell, & Malaiya, Yashwant K. (1992a). Prediction of software reliability using connectionist models. *IEEE Transactions on Software Engineering, 18*(7), 563-574.

Karunanithi, Nachimuthu, Whitley, Darrell, & Malaiya, Yashwant K. (1992b). Using neural networks in reliability prediction. *IEEE Software, 9*(4), 53-59.

Lakshmanan, Indhurani, & Ramasamy, Subburaj. (2015). An artificial neural-network approach to software reliability growth modeling. *Procedia Computer Science, 57*, 695-702.

Lo, Jung-Hua. (2009). *The implementation of artificial neural networks applying to software reliability modeling.* Paper presented at the 2009 Chinese Control and Decision Conference.

Lyu, Michael R. (1996). *Handbook of software reliability engineering* (Vol. 222): IEEE computer society press CA.

Mallikharjuna Rao, K, & Anuradha, K. (2016). A New Method to Optimize the Reliability of Software Reliability Growth Models using Modified Genetic Swarm Optimization. *International Journal of Computer Applications, 145*(5).

Mallikharjuna, Rao K., & Kodali, Anuradha. (2015). *An Efficient Method for Parameter Estimation of Software Reliability Growth Model Using Artificial Bee Colony Optimization*, Cham.

Mir, Khurshid Ahmad. (2011). A software reliability growth model. *Journal of Modern Mathematics and Statistics, 5*(1), 13-16.

MirRokni, Mahshid Kaviani1 Seyed Majid. (2017). Applying genetic algorithm in architecture and neural network training. *International Journal of Computer Science and Network Security IJCSN, 17*(6), 118.

Montana, David J, & Davis, Lawrence. (1989). *Training Feedforward Neural Networks Using Genetic Algorithms.* Paper presented at the IJCAI.

Nielsen, Michael A. (2015). *Neural networks and deep learning* (Vol. 25): Determination press San Francisco, CA, USA:.

Ong, Liang Fuh, Isa, Mohd Adham, Jawawi, Dayang NA, & HALIM, HAHLIZA ABDUL. (2017). IMPROVING SOFTWARE RELIABILITY GROWTH MODEL SELECTION RANKING USING PARTICLE SWARM OPTIMIZATION. *Journal of Theoretical & Applied Information Technology, 95*(1).

Rahman, Md Mijanur, & Setu, Tania Akter. (2015). An implementation for combining neural networks and genetic algorithms. *Int. J. Comp. Sci. Technol, 6*, 218-222.

Ramasamy, Subburaj, & Lakshmanan, Indhurani. (2016). Application of artificial neural network for software reliability growth modeling with testing effort. *Indian Journal of Science and Technology, 9*(29).

Saleem, Nada N. (2013). Software reliability prediction using artificial techniques. *IJCSI, 10*(4, No 2).

Singh, Yogesh, & Kumar, Pradeep. (2010). *Prediction of software reliability using feed forward neural networks.* Paper presented at the 2010 International Conference on Computational Intelligence and Software Engineering.

Su, Yu-Shen, & Huang, Chin-Yu. (2007). Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. *Journal of Systems and Software, 80*(4), 606-615.

Tawfiq, Luma NM, & Salih, Othman M. (2014). *Using Feed Forward Neural Network to Solve Eigenvalue Problems.* Paper presented at the Conference Papers in Science.

Wang, Gaozu, & Li, Weihuai. (2010). *Research of software reliability combination model based on neural net.* Paper presented at the 2010 Second World Congress on Software Engineering.

Wu, Wei, Han, Kun, He, Chengming, & Wu, Shujian. (2012). *A dynamically-weighted software reliability combination model.* Paper presented at the 2012 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering.

Zheng, Jun. (2009). Predicting software reliability with neural network ensembles. *Expert systems with applications, 36*(2), 2116-2122.

Appendix A
Datasets

The data set comes from [Musa 79c] and Brazilian Electronic Switching System. Assembly Language [Kano93b, Mart91, Lapr91] and Real-Time Command & Control System from Software project failure datasets

Dataset 2:

Cumulative Execution Time	Cumulative Failure Number		Cumulative Execution Time	Cumulative Failure Number
1	60		200	212
2	30		201	4
3	540		202	5
4	67		203	106
5	40		204	264
6	23		205	269
7	5		206	276
8	53		207	1
9	4		208	203
10	16		209	117
11	94		210	1
12	15		211	45
13	5		212	5
14	90		213	110
15	77		214	18
16	68		215	10
17	15		216	179
18	137		217	66
19	23		218	1
20	1		219	106
21	104		220	2

22	16		221	19
23	9		222	117
24	10		223	30
25	12		224	130
26	4		225	31
27	10		226	28
28	82		227	4
29	6		228	302
30	54		229	362
31	25		230	5
32	43		231	63
33	12		232	42
34	48		233	86
35	23		234	258
36	6		235	294
37	10		236	256
38	12		237	118
39	14		238	13
40	33		239	47
41	9		240	92
42	4		241	343
43	66		242	128
44	0.5		243	392
45	18		244	90
46	15		245	116
47	75		246	35
48	30		247	171
49	116		248	139
50	14		249	110
51	15		250	98
52	41		251	60

53	1		252	90
54	99		253	82
55	9		254	5
56	45		255	30
57	68		256	35
58	36		257	231
59	50		258	62
60	81		259	158
61	89		260	1622
62	85		261	353
63	54		262	33
64	3		263	70
65	15		264	35
66	6		265	116
67	8		266	809
68	36		267	1710
69	98		268	745
70	32		269	350
71	36		270	470
72	5		271	122
73	9		272	244
74	60		273	2384
75	34		274	249
76	16		275	607
77	164		276	83
78	123		277	2
79	19		278	26
80	19		279	586
81	126		280	352
82	36		281	673
83	54		282	330

84	15		283	649
85	16		284	123
86	154		285	1789
87	84		286	1288
88	92		287	111
89	247		288	75
90	244		289	74
91	60		290	333
92	5		291	287
93	2		292	1
94	5		293	881
95	130		294	13
96	0.5		295	1314
97	233		296	472
98	50		297	363
99	54		298	6
100	52		299	4
101	57		300	55
102	1		301	409
103	2		302	36
104	5		303	15
105	1		304	1404
106	17		305	17
107	14		306	71
108	2		307	34
109	87		308	434
110	19		309	60
111	29		310	19
112	0.5		311	20
113	61		312	79
114	118		313	24

115	20		314	540
116	3		315	1040
117	11		316	38
118	87		317	78
119	5		318	41
120	249		319	1757
121	28		320	205
122	44		321	2095
123	31		322	788
124	3		323	1
125	10		324	2668
126	3		325	470
127	8		326	10
128	17		327	20
129	3		328	338
130	55		329	222
131	7		330	28
132	12		331	56
133	6		332	561
134	4		333	65
135	169		334	100
136	30		335	900
137	4		336	212
138	38		337	287
139	7		338	53
140	4		339	3
141	4		340	4973
142	13		341	197
143	10		342	1174
144	40		343	783
145	57		344	1346

146	22		345	59
147	37		346	98
148	127		347	1594
149	12		348	25
150	8		349	98
151	1		350	722
152	21		351	228
153	104		352	78
154	8		353	33
155	23		354	453
156	1		355	1020
157	6		356	4327
158	141		357	925
159	3		358	302
160	21		359	649
161	3		360	43
162	18		361	185
163	0.5		362	157
164	75		363	30
165	92		364	1771
166	39		365	1088
167	29		366	556
168	3		367	55
169	2		368	4892
170	158		369	81
171	30		370	61
172	24		371	476
173	5		372	63
174	92		373	3
175	7		374	3
176	33		375	62

177	20		376	1
178	16		377	44
179	292		378	1236
180	3		379	1406
181	9		380	109
182	12		381	1471
183	18		382	1797
184	9		383	1749
185	75		384	2096
186	15		385	76
187	46		386	2167
188	9		387	2059
189	94		388	2177
190	25		389	1893
191	175		390	198
192	5		391	3326
193	12		392	3100
194	18		393	586
195	70		394	2686
196	3		395	124
197	10		396	229
198	114		397	1008
199	213			

```
% INITIALIZE THE NEURAL NETWORK PROBLEM %

% inputs for the neural net(AND gate example== 2 inputs && 4 samples)
inputs = xlsread('inputs.xlsx')';

% targets for the neural net
targets = xlsread('targets.xlsx')';

% number of neurons
n = 3;

% create a neural network
net = feedforwardnet(n);

% configure the neural network for this dataset
net = configure(net, inputs, targets);
% get the normal NN weights and bias
getwb(net)

% error MSE normal NN
error = targets - net(inputs);
RMSE = mean(error.^2)/mean(var(targets',1))
% create handle to the MSE_TEST function, that
% calculates MSE
h = @(x) NMSE(x, net, inputs, targets);

% Setting the Genetic Algorithms tolerance for
% minimum change in fitness function before
% terminating algorithm to 1e-8 and displaying
% each iteration's results.

ga_opts = gaoptimset('PopInitRange', [0;1], 'TolFun', 1e-
10,'display','iter','PopulationSize',100,'FitnessScalingFcn',@fitscalingprop,
'SelectionFcn',@selectiontournament,'CrossoverFcn',@crossoverscattered,'Mutat
ionFcn',@mutationgaussian);
ga_opts = gaoptimset(ga_opts, 'StallGenLimit', 100, 'FitnessLimit', 1e-5,
'Generations', 100);
% PLEASE NOTE: For a feed-forward network
% with n hidden neurons, 3n+n+1 quantities are required
% in the weights and biases column vector.
% a. n for the input weights=(features*n)=3*n
% b. n for the input biases=(n bias)=n
% c. n for the output weights=(n weights)=n
% d. 1 for the output bias=(1 bias)=1
% running the genetic algorithm with desired options
% Y'(t)= W21(1- e-W11t) + W22/ (1 + b1e-W12t) + W23(1-(1+W13t) e-W13t)).
[x, err_ga] = ga(h, 3*n+n+n+1, ga_opts);
net = setwb(net, x');
% get the GA optimized NN weights and bias
getwb(net)

% error MSE GA optimized NN
error = targets - net(inputs);
RMSE = mean(error.^2)/mean(var(targets',1))
```