ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

An Efficient Method for Software Reliability Using Modified Genetic Algorithm: Inflection S-Shaped Model

[1] Dr. R.Satyaprasad, [2] G. Bharathi
[1] Assoc.Professor, Dept. of CSE, Acharya Nagarjuna University.
[2] Research Scholar, Rayalaseema University.

Abstract - **In order to assess software reliability, many software reliability growth models (SRGMs) have been proposed in the past four decades. In principle, two widely used methods for the parameter estimation of SRGMs are the maximum likelihood estimation (MLE) and the least squares estimation (LSE). However, the approach of these two estimations may impose some restrictions on SRGMs, such as the existence of derivatives from formulated models or the needs for complex calculation. In this paper, we propose a modified genetic algorithm (MGA) to assess the reliability of software considering the Time domain software failure data using Inflection S-shaped model which is NonHomogenous Poisson Process (NHPP) based. Experiments based on real software failure data are performed, and the results show that the proposed genetic algorithm is more effective and faster than traditional algorithms.**

Keywords - Software reliability, Inflection S-shaped model, Time domain data, Mean Value Function, Modified Genetic Algorithm, NHPP.

I. INTRODUCTION

Software reliability assessment is important to evaluate the quality of software system, since it is one of the most important attribute of software. One of the most difficult problems of software industry is to ship a reliable product. Therefore it is necessary to have accurate and fast estimation techniques for verifying software reliability. For Four decades, many Software Reliability Growth Models (SRGMs) have been proposed in estimating reliability growth of software products. SRGMs can be used to depict the behaviour of observed software failures characterized by either times of failures (i.e Time domain data) or by the number of failures at fixed times (i.e Interval domain data) (Lyu, 1996).

The parameters of SRGMs are generally unknown and have to be estimated based on collected failure data. Two of the most popular estimation techniques are Maximum Likelihood Estimation (MLE) and Least Squares Estimation (LSE) (Goel, 1985; Ohba, 1984). In fact, MLE and LSE involve the property of probability theory and statistical analysis. Thus, this may impose some restrictions on the parameter estimation of SRGMs (Costa *et al.*, 2007; Minohara and Tohma, 1995) such as the continuity, the unimodality, the existence of derivatives from formulated models, the complex likelihood function, etc. The method of MLE estimation by solving a set of simultaneous equations and is better in deriving confidence intervals. The method of LSE minimizes the sum of squares of the deviations between what we actually observe and what we expect. Nevertheless, LSE is suitable for fitting data from small to medium sample sizes (Wood, 1996), while MLE is considered to be better statistical estimator for large sample sizes. In particular, when the formulated model of SRGMs is complicated or the sample size of failure data is large, these two estimation techniques may not be effective to find out the optimal solutions and generally require to be solved numerically. Hence, the more effective and applicable approaches for the parameter estimation of SRGMs may be necessary. In recent years, the Genetic Algorithms (GAs) has gained popularity in solving the optimization problem of scientific fields (Goldberg, 1989 ; Mitchell, 1998). Because, the parameter estimation can be reformulated as a searching process within the domain of all the feasible solutions (Harman and Jones, 2001; Jiang, 2006), it may be attractive to introduce GA into the process of software reliability modeling (Dai *et al.*, 2003). Therefore, in this paper we will propose a modified genetic algorithm (MGA) to estimate the parameter of the SRGMs. We will attempt to modify GA's operators with weighted bit mutation and a rebuilding mechanism to improve the performance and efficiency of estimations. Finally, the applicability of proposed MGA, the result of parameter estimation and the reliability with Inflection S-shaped model will also be demonstrated through real data.

The rest of this paper is organized as follows. Section 2 surveys NHPP based SRGMs and in specific Inflection S-

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 10, November 2017

shaped Model along with the past researches of GAs in software engineering areas. In Section 3, an effective MGA is proposed to solve the parameter estimation of reliability models. Then, the experimental results based on two failure data are presented and discussed in Section 4. Finally, some conclusions are given in Section 5.

II. LITERATURE SURVEY.

A. NHPP model.

The Non-Homogenous Poisson Process (NHPP) based software reliability growth models (SRGMs) are proved to be quite successful in practical software reliability engineering (Musa et al., 1987). The main issue in the NHPP model is to determine an appropriate mean value function to denote the expected number of failures experienced up to a certain time point. Model parameters can be estimated by using Modified Genetic Algorithm (MGA). Various NHPP SRGMs have been built upon various assumptions. Many of the SRGMs assume that each time a failure occurs, the fault that caused it can be immediately removed and no new faults are introduced. Which is usually called perfect debugging. Imperfect debugging models have proposed a relaxation of the above assumption (Pham, 1993).

A fault is a statement in a program which causes one or more failures. Software Reliability Growth is defined by the mathematical relationship that exists between the time span of testing a program and the cumulative number of errors discovered. After failure detection, we find a fault and define a fix for the fault. The exponential software reliability growth models are designed to describe the failure detection process.

Let $\{N(t), t \geq 0\}$ be the cumulative number of software failures by time 't'. m(t) is the mean value function, representing the expected number of software failures by time 't'. $\lambda(t)$ is the failure intensity function, which is proportional to the residual fault content. Thus $m(t) = a\left(1-e^{-bt}\right)$ and $\lambda(t) = \frac{dm(t)}{dt}$. where 'a' denotes the initial number of faults contained in a program and 'b' represents the fault detection rate. In software reliability, the initial number of faults and the fault detection rate are always unknown. The maximum likelihood technique can be used to evaluate the unknown parameters. This paper deals with the application of Inflection S-shaped model on application test data collected from literature, which is of Time domain data (i.e ungrouped).

SRGMs are a statistical interpolation of defect detection data by mathematical functions. They have been grouped into two classes of models-Concave and S-shaped. The only way to verify and validate the software is by testing. This involves running the software and checking for unexpected behaviour of the software output (kapur, 2009). SRGMs are used to estimate the reliability of a software product. In literature, we have several SRGMs developed to monitor the reliability growth during the testing phase of the software development. Software reliability is defined as the probability of failure-free software operation for specified period of time 't' in a specified environment.

Inflection S-shaped model

Software reliability growth models(SRGM's) are useful to assess the reliability for quality management and testing-progress control of software development. They have been grouped into two classes of models concave and S-shaped. The most important thing about both models is that they have the same asymptotic behavior, i.e., the defect detection rate decreases as the number of defects detected (and repaired) increases, and the total number of defects detected asymptotically approaches a finite value. The inflection S-shaped model was proposed by Ohba in 1984. This model assumes that the fault detection rate increases throughout a test period. The model has a parameter, called the inflection rate, that indicates the ratio of detectable faults to the total number of faults in the target software. True, sustained exponential growth cannot exist in the real world. Eventually all exponential, amplifying processes will uncover underlying stabilizing processes that act as limits to growth. The shift from exponential to asymptotic growth is known as sigmoidal, or S-shaped, growth.

Ohba models the dependency of faults by postulating the following assumptions:

- Some of the faults are not detectable before some other faults are removed.
- The detection rate is proportional to the number of detectable faults in the program.
- Failure rate of each detectable fault is constant and identical.
- All faults can be removed.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 10, November 2017

Assuming [Ohba 1984b]: $b(t) = \dfrac{b}{1+\beta e^{-bt}}$

This model is characterized by the following mean value function:

$$m(t) = \frac{a}{1+\beta e^{-bt}}\left(1-e^{-bt}\right)$$

where 'b' is the failure detection rate, and 'β' is the inflection factor. The failure intensity function is given as:

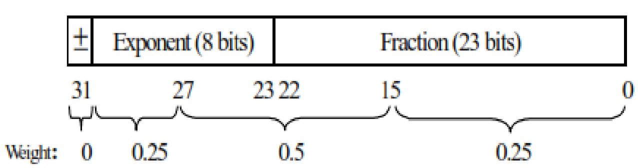
$$\lambda(t) = \frac{abe^{-bt}(1+\beta)}{\left(1+\beta e^{-bt}\right)^2}.$$

III. MIDIFIED GENETIC ALGORITHM.

Genetic Algorithm (GA) has been popularly used to solve various optimization problems. GA has advantages of easy implementation with large search space and rapid convergence on good quality solutions. It does not impose restrictions on the continuity, the existence of derivatives, and the unimodality of evaluation functions. Traditional GA has several steps for searching process:

- **chromosome representation;**

GA simulates the initial population of parametric solution represented as chromosomes. Each chromosome is encoded as string of bits. Since the parameters of SRGMs are usually real numbers, we proposed an IEEE floating-point standard to encode chromosomes.



Chromosome Representation and Weighted Bit Mutation

- **fitness function;**
    - ***least squares estimation*** **(LSE)**

$$\text{fitness\_LSE} = \frac{1}{\text{MSE}}$$

Where, MSE is a measure to compare the differences between actual values and estimators

- **Selection scheme:** The selection scheme is to select the candidate chromosomes from the current population based on their fitness values. Our goal is to maximize fitness function for finding the best parameters. With these fitness values, we can further adopt roulette wheel selection and uniform crossover to choose candidate chromosomes. The roulette wheel selection does not guarantee that the fittest chromosome will always be selected for generating offspring in searching process. This may spend more numbers of generations on finding a solution. Thus, we propose a rebuilding mechanism. Among each generation, one best chromosome is kept at the end of the population to avoid disappearance from the selection scheme. This mechanism does not violate GA's original purpose. If the next generation produces a much more suitable chromosome, the previous kept chromosome will be replaced.
- **Crossover operator:** Two chromosomes are chosen from the population and are exchanged in part with each other in order to improve their fitness value. The uniform crossover is one of the simplest form (Goldberg, 1989). The crossover may happen at different bits with a probability called crossover rate, P. This rate typically ranges from 0.5 to 0.8 from GA literatures (Jiang, 2007). It is decide to adopt uniform crossover in our experiments.
- **Mutation operator:** We found that some bits are less efficient during bit mutation based on IEEE floating-point format. If we mutate at sign bit, the whole string will be changed from a plus to a minus. Because the estimated parameters are usually a positive real number, this mutation may be useless. Similarly, if we mutate at a very high exponential bit or at a very low fractional bit, the whole string will respectively be $2^{\pm128}$ times the original or only be changed slightly. In fact, these mutations may be too severe or negligible. **Sensitivity analysis** on different bit mutations will be investigated. Depending on this analytic result, we further provide a weighted bit mutation.
- **Stopping criteria:** The searching process will iteratively evolve parametric solutions until the maximal generations equal to 10000 trials or the best fitness function does not change in the past 1000 trials.

A. Algorithm for parameter estimation

In this section, we show how to modify the traditional GA to estimate the parameters of SRGMs. The detailed algorithm of MGA is shown below. It is noted that all the proposed mechanisms of MGA are built by using Java programming language.

1. Initialize a population of chromosomes randomly

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 10, November 2017

2. FOR (Iteration i=1; i<=Maximum generation && termination condition=FALSE; i=i+1)
    a. Calculate fitness for all individual chromosomes
    b. Reproduce offspring by roulette selection
    c. Choose two chromosomes from the population in order and randomize a probability p
    d. IF p < Crossover rate THEN
        i. Generate two offsprings by recombining two chromosomes.
       ENDIF
    e. Choose a chromosome from the population in order and randomize a probability q
    f. IF q < Mutation rate THEN
        i. mutate the chosen chromosome at a weighted bit position
       ENDIF
    g. Keep the fittest parent in the end of population
    h. Check termination condition
3. ENDFOR
4. Output estimated parameters

I. ILLUSTRATING THE MGA.

A. Data Analysis.

There are two common types of failure data: time-domain and interval-domain. Some software reliability models can handle both types of data. The time domain approach involves recording the individual times at which failure occurred. The interval domain approach is characterized by counting the number of failures occurring during a fixed period (e.g., test session, hour, week, day). The collected data is the Time Between Failures. Based on the failure data collected from the literature, we used cumulative failures data for software reliability using Inflection S-shaped model.

B. Distribution of Time between failures

Based on the inter failure data given in Table 2 and 3, we compute the software failures process through Mean Value Control chart. We used cumulative time between failures data for software reliability monitoring using Inflection S-shaped model. The use of cumulative quality is a different and new approach, which is of particular advantage in reliability.

'$\hat{a}$' and '$\hat{b}$' are Maximum Likely hood Estimates of parameters and the values can be computed using iterative method for the given cumulative time between failures data. Using 'a' and 'b' values we can compute $m(t)$.

Assuming an acceptable probability of false alarm of 0.27%, the control limits can be obtained as (Xie, 2002):

$$T_U = \frac{1}{1+\beta e^{-bt}}\left(1-e^{-bt}\right) = 0.99865$$

$$T_C = \frac{1}{1+\beta e^{-bt}}\left(1-e^{-bt}\right) = 0.5$$

$$T_L = \frac{1}{1+\beta e^{-bt}}\left(1-e^{-bt}\right) = 0.00135$$

These limits are converted to $m(t_U)$, $m(t_C)$ and $m(t_L)$ form. They are used to find whether the software process is in control or not by placing the points in Failure control shown in figure 1 and 2 . A point below the control limit $m(t_L)$ indicates an alarming signal. A point above the control limit $m(t_U)$ indicates better quality. If the points are falling within the control limits, it indicates the software process is in stable condition. The values of control limits are as follows.

TABLE I. Estimated parameters and control limits

Data Set	Parameters		Control limits		
	a	B	UCL	CL	LCL
DS1	84.963130	0.039024	84.848429	42.481564	0.114700
DS2	22.000137	0.093427	21.970437	11.000068	0.029700

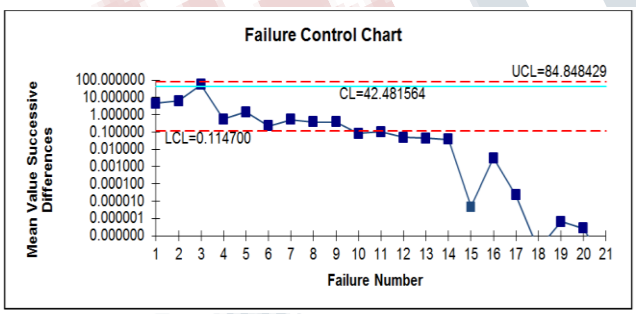
TABLE II. DS1 - Successive differences of mean value function

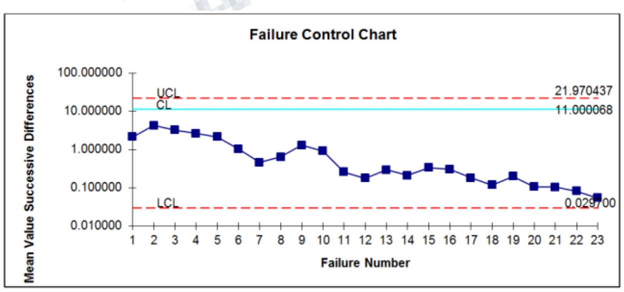
Failure Number	Time Between failures	m(t)	Successive Differences
1	5.5	15.775220	4.596044
2	7.33	20.371264	6.358231
3	10.08	26.729496	54.455937
4	80.97	81.185432	0.537410

5	84.91	81.722843	1.432892
6	99.89	83.155735	0.228695
7	103.36	83.384430	0.508112
8	113.32	83.892542	0.384027
9	124.71	84.276569	0.370446
10	144.59	84.647015	0.083036

TABLE III. DS2 - Successive differences of mean value function

Failure Number	Time Between failures	m(t)	Successive Differences
1	0.5	0.958339	2.140314
2	1.7	3.098653	4.212373
3	4.5	7.311025	3.193476
4	7.2	10.504501	2.595063
5	10	13.099564	2.143737
6	13	15.243301	1.032924
7	14.8	16.276225	0.456349
8	15.7	16.732574	0.639359
9	17.1	17.371933	1.281512
10	20.6	18.653445	0.905970





IV. CONCLUSION.

A number of estimates of software quality are based on the parameter estimates of SRGMs. Therefore, the quality estimates can be derived based the quality estimates of parameters. Inorder to estimate the Software reliability, a robust method of estimating parameter is employed on Interval domain software failure data. The results of software reliability over the two failure data sets with Inflection S-shaped model is presented in table II and Table III.

REFERENCES

[1] Costa, E. O., de Souza, G. A.., Pozo, A. T. R and Vergilio, S. R. (2007). "Exploring Genetic Programming and Boosting Techniques to Model Software Reliability," IEEE Transactions on Reliability, vol.56, no. 3, pp. 422-434.

[2] Dai, Y. S. Poh, K. L and Yang, B. (2003). "Optimal Testing-Resource Allocation with Genetic Algorithm for Modular Software Systems," Journal of Systems and Software, vol. 66, no. 1, pp. 47-55.

[3] Goel, A. L. (1985). "Software Reliability Models: Assumptions, Limitations, and Applicability," IEEE Transactions on Software Engineering, vol. 11, no. 12, pp. 1411-1423.

[4] Goldberg, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[5] Jiang, H. Y. (2006). "Can the Genetic Algorithm Be a Good Tool for Software Engineering Searching Problems?," Proceedings of the 30th IEEE International Computer Software and Applications Conference (COMPSAC 2006), pp. 362-366, Chicago, USA.

[6] Kapur, P.K., Sunil kumar, K., Prashant, J. Ompal, S. (2009). "Incorporating concept of two types of imperfect debugging for developing flexible software reliability growth model in distributed development environment", Journal of Technology and Engineering sciences, Vol.1, No.1; Jan-Jun.

[7] Lyu, M. R. Handbook of Software Reliability Engineering, McGraw-Hill, 1996.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 10, November 2017

[8] M. Harman and B. F. Jones, "Search-Based Software Engineering," Information and Software Technology, vol. 43, no. 14, pp. 833-839, 2001.

[9] Minohara, T and Tohma, Y. (1995). "Parameter Estimation of Hyper-Geometric Distribution Software Reliability Growth Model by Genetic Algorithms," Proceedings of the 6th IEEE International Symposium on Software Reliability Engineering (ISSRE 1995), pp. 324-329, Toulouse, France.

[10] Mitchell, M. An Introduction to Genetic Algorithms, The MIT Press, 1998.

[11] Musa, J.D., Iannino, A., Okumoto, k., 1987. “Software Reliability: Measurement Prediction Application”. McGraw-Hill, New York.

[12] Ohba, M. (1984). "Software Reliability Analysis Models," IBM Journal of Research and Development, vol. 28, no. 4, pp. 428-443.

[13] Pham. H., 1993. “Software reliability assessment: Imperfect debugging and multiple failure types in software development”. EG&G-RAAM10737; Idaho National Engineering Laboratory.

[14] Wood, A. (1996). "Predicting Software Reliability," IEEE Computer, vol. 29, no. 11, pp. 69-77.

[15] Xie, M., Goh. T.N., Ranjan.P., “Some effective control chart procedures for reliability monitoring” - Reliability engineering and System Safety 77 143 -150, 2002.