IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 5, Issue 5, Oct - Nov, 2017
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

Modified Genetic Algorithm Based Software Reliability Using SPRT: GO Model

U.Usha Rani, Dr. R.Satyaprasad

Research Scholar, Rayalaseema University.
Assoc.Professor, Dept. of CSE, Acharya Nagarjuna University.

In Classical Hypothesis testing volumes of data is to be collected and then the conclusions are drawn, which may need more time. But, Sequential Analysis of Statistical science could be adopted in order to decide upon the reliability or unreliability of the developed software very quickly. The procedure adopted for this is, Sequential Probability Ratio Test (SPRT). It is designed for continuous monitoring. The likelihood based SPRT proposed by Wald is very general and it can be used for many different probability distributions. In the present paper we propose the performance of SPRT on 4 data sets of Time domain data using exponential model and analyzed the results. The parameters are estimated using Maximum Likelihood Estimation method.

1. INTRODUCTION

Sequential analysis is a method of statistical inference whose main feature is that the number of observations required by the procedure is not determined in advance. The decision to end the observations depends, at each stage, on the results of the samples already taken. (SPRT), which is usually applied in situations, requires a decision between two simple hypothesis or a single decision point. Wald's (1947) SPRT procedure has been used to classify the software under test into one of two categories (e.g., reliable/unreliable, pass/fail, certified/noncertified) (Reckase, 1983). Wald's procedure is particularly relevant if the data is collected sequentially. Classical Hypothesis Testing is different from Sequential

Analysis. In Classical Hypothesis testing, the number of cases tested or collected is fixed at the beginning of the experiment. In this method, the analysis is made and conclusions are drawn after collecting the complete data.

In the analysis of software failure data, either TBFs or failure count in a given time interval is dealt with. If it is further assumed that the average number of recorded failures in a given time interval is directly proportional to the length of the interval and the random number of failure occurrences in the interval is explained by a Poisson process. Then it is known that the probability equation of the stochastic process representing the failure occurrences is given by a Homogeneous Poisson Process with the expression

$$P\left[N(t)=n\right]=\frac{e^{-\lambda t}(\lambda t)^{n}}{n!} \qquad (1.1)$$

Stieber (1997) observes that, the application of SRGMs may be difficult and reliability predictions can be misleading, if classical testing strategies are used. However, he observes that statistical methods can be successfully applied to the failure data. He demonstrated his observation by applying the well-known sequential probability ratio test of Wald for a software failure data to detect unreliable software components and compare the reliability of different software versions. In this chapter the popular SRGM – Exponential model is considered and the principle of Stieber is adopted in detecting unreliable software in order to accept or reject the developed software. The theory proposed by Stieber is presented in Section 2 for a ready reference. Extension of this theory to the considered SRGM is presented in Section 3. Modified Genetic Algorithm based parameter estimation method is presented in Section 4. Application of the decision rule to detect unreliable software with reference to the SRGM is given in Section 5.

2. SEQUENTIAL TEST FOR A POISSON PROCESS

A.Wald, developed the SPRT at Columbia University in 1943. A big advantage of sequential tests is that they require fewer observations (time) on the average than fixed sample size tests. SPRTs are widely used for statistical quality control in manufacturing processes. The SPRT for Homogeneous Poisson Processes is described below.

Let $\{N(t), t \geq 0\}$ be a homogeneous Poisson process with rate ' λ '. In this case, $N(t)$ = number of failures up to time ' t' and ' λ ' is the failure rate (failures per unit time). If the system is put on test (for example a software system, where testing is done according to a usage profile and no faults are corrected) and that if we want to estimate its failure rate ' λ '. We can not expect to estimate ' λ ' precisely. But we want to reject the system with a high probability if the data suggest that the failure rate is larger than λ_1 and accept it with a high probability, if it is smaller than λ_0. As always with statistical tests, there is some risk to get the wrong answers. So we have to specify two (small) numbers ' α ' and ' β ', where ' α ' is the probability of falsely rejecting the system. That is rejecting the system even if $\lambda \leq \lambda_0$. This is the "producer's" risk. 'β' is the probability of falsely accepting the system .That is accepting the system even if $\lambda \leq \lambda_1$. This is the "consumer's" risk. Wald's classical SPRT is very sensitive to the choice of relative risk required in the specification of the alternative hypothesis. With the classical SPRT, tests are performed continuously at every time point $t > 0$ as additional data are collected. With specified choices of λ_0 and λ_1 such that $0 < \lambda_0 < \lambda_1$, the probability of finding $N(t)$ failures in the time span $(0, t)$ with λ_1, λ_0 as the failure rates are respectively given by

$$P_1 = \frac{e^{-\lambda_1 t}\left[\lambda_1 t\right]^{N(t)}}{N(t)!} \qquad (2.1)$$

$$P_0 = \frac{e^{-\lambda_0 t}\left[\lambda_0 t\right]^{N(t)}}{N(t)!} \qquad (2.2)$$

The ratio $\frac{P_1}{P_0}$ at any time 't' is considered as a measure of deciding the truth towards λ_0 or λ_1, given a sequence of time instants say $t_1 < t_2 < t_3 < ..... < t_K$ and the corresponding realizations $N(t_1), N(t_2), ...N(t_K)$ of $N(t)$. Simplification of $\frac{P_1}{P_0}$ gives

$$\frac{P_1}{P_0} = \exp(\lambda_0 - \lambda_1)t + \left(\frac{\lambda_1}{\lambda_0}\right)^{N(t)}$$

The decision rule of SPRT is to decide in favour of λ_1, in favour of λ_0 or to continue by observing the number of failures at a later time than 't' according as $\frac{P_1}{P_0}$ is greater than or equal to a constant say A, less than or equal to a constant say B or in between the constants A and B. That is, we decide the given software product as unreliable, reliable or continue (Satyaprasad, 2007) the test process with one more observation in failure data, according to

$$\frac{P_1}{P_0} \geq A \qquad (2.3)$$

$$\frac{P_1}{P_0} \leq B \qquad (2.4)$$

$$B < \frac{P_1}{P_0} < A \qquad (2.5)$$

The approximate values of the constants A and B are taken as $A \cong \frac{1-\beta}{\alpha}$, $B \cong \frac{\beta}{1-\alpha}$

Where 'α' and 'β' are the risk probabilities as defined earlier. A good test is one that makes the α and β errors as small as possible. The common procedure is to fix the β error

and then choose a critical region to minimize the error or maximize the power i.e $1-\beta$ of the test. A simplified version of the above decision processes is to reject the system as unreliable if $N(t)$ falls for the first time above the line

$$N_U(t) = a.t + b_2 \tag{2.6}$$

To accept the system to be reliable if $N(t)$ falls for the first time below the line

$$N_L(t) = a.t - b_1 \tag{2.7}$$

To continue the test with one more observation on $(t, N(t))$ as the random graph of $[t, N(t)]$ is between the two linear boundaries given by equations (2.6) and (2.7) where

$$a = \frac{\lambda_1 - \lambda_0}{\log\left(\frac{\lambda_1}{\lambda_0}\right)} \tag{2.8}$$

$$b_1 = \frac{\log\left[\frac{1-\alpha}{\beta}\right]}{\log\left(\frac{\lambda_1}{\lambda_0}\right)} \tag{2.9}$$

$$b_2 = \frac{\log\left[\frac{1-\beta}{\alpha}\right]}{\log\left(\frac{\lambda_1}{\lambda_0}\right)} \tag{2.10}$$

The parameters α, β, λ_0 and λ_1 can be chosen in several ways. One way suggested by Stieber is $\lambda_0 = \frac{\lambda.\log(q)}{q-1}$, $\lambda_1 = q\frac{\lambda.\log(q)}{q-1}$ *where* $q = \frac{\lambda_1}{\lambda_0}$

If λ_0 and λ_1 are chosen in this way, the slope of $N_U(t)$ and $N_L(t)$ equals λ. The other two ways of choosing λ_0 and λ_1 are from past projects (for a comparison of the projects) and from part of the data to compare the reliability of different functional areas.

3. SEQUENTIAL TEST FOR SOFTWARE RELIABILITY GROWTH MODELS

In Section 2, for the Poisson process it is known that the expected value of $N(t) = \lambda t$ called the average number of failures experienced in time 't' .This is also called the mean value function of the Poisson process. On the other hand if we consider a Poisson process with a general function (not necessarily linear) $m(t)$ as its mean value function the probability equation of a such a process is

$$P[N(t) = Y] = \frac{[m(t)]^y}{y!}.e^{-m(t)}, y = 0,1,2,----$$

Depending on the forms of $m(t)$ various Poisson processes called NHPP are obtained. For our two parameter Exponential model, the mean value function is given as $m(t) = a\left(1-e^{-(bt)}\right)$ where $a > 0, b > 0$

It may be written as

$$P_1 = \frac{e^{-m_1(t)}.[m_1(t)]^{N(t)}}{N(t)!}$$

$$P_0 = \frac{e^{-m_0(t)}.[m_0(t)]^{N(t)}}{N(t)!}$$

Where, $m_1(t)$, $m_0(t)$ are values of the mean value function at specified sets of its parameters indicating reliable software and unreliable software respectively. Let P_0, P_1 be

values of the NHPP at two specifications of b say b_0, b_1, where $(b_0 < b_1)$. It can be shown that for our model $m(t)$ at b_1 is greater than that at b_0. Symbolically $m_0(t) < m_1(t)$. Then the SPRT procedure is as follows:

Accept the system to be reliable if, $\frac{P_1}{P_0} \le B$

i.e., $\frac{e^{-m_1(t)}.[m_1(t)]^{N(t)}}{e^{-m_0(t)}.[m_0(t)]^{N(t)}} \le B$

i.e., $N(t) \le \frac{\log\left(\frac{\beta}{1-\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)}$ (3.1)

Decide the system to be unreliable and reject if, $\frac{P_1}{P_0} \ge A$

i.e., $N(t) \ge \frac{\log\left(\frac{1-\beta}{\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)}$ (3.2)

Continue the test procedure as long as

$$\frac{\log\left(\frac{\beta}{1-\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} < N(t) < \frac{\log\left(\frac{1-\beta}{\alpha}\right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} \quad (3.3)$$

Substituting the appropriate expressions of the respective mean value function $-m(t)$ of Exponential we get the respective decision rules and are given in following lines

Acceptance region:

$$N(t) \le \frac{\log\left(\frac{\beta}{1-\alpha}\right) + a\left(e^{-(b_0 t)} - e^{-(b_1 t)}\right)}{\log\left(\frac{1-e^{-(b_1 t)}}{1-e^{-(b_0 t)}}\right)} \quad (3.4)$$

Rejection region:

$$N(t) \geq \frac{\log\left(\frac{1-\beta}{\alpha}\right) + a\left(e^{-(b_0 t)} - e^{-(b_1 t)}\right)}{\log\left(\frac{1-e^{-(b_1 t)}}{1-e^{-(b_0 t)}}\right)} \qquad (3.5)$$

Continuation region:

$$\frac{\log\left(\frac{\beta}{1-\alpha}\right) + a\left(e^{-(b_0 t)} - e^{-(b_1 t)}\right)}{\log\left(\frac{1-e^{-(b_1 t)}}{1-e^{-(b_0 t)}}\right)} < N(t) < \frac{\log\left(\frac{1-\beta}{\alpha}\right) + a\left(e^{-(b_0 t)} - e^{-(b_1 t)}\right)}{\log\left(\frac{1-e^{-(b_1 t)}}{1-e^{-(b_0 t)}}\right)} \qquad (3.6)$$

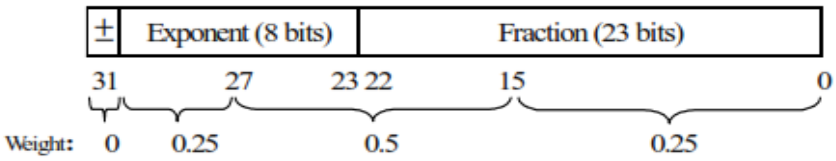
It may be noted that in the above mentioned model the decision rules are exclusively based on the strength of the sequential procedure (α, β) and the values of the respective mean value functions namely, $m_0(t)$, $m_1(t)$. If the mean value function is linear in 't' passing through origin, that is, $m(t) = \lambda t$ the decision rules become decision lines as described by Stieber. In that sense equations (3.1), (3.2), (3.3) can be regarded as generalizations to the decision procedure of Stieber. The applications of these results for live software failure data are presented with analysis in Section 5.

4. 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:** This 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. Arebuilding mechanism is proposed. 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.
- **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 forms (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, 2006). It is decide to adopt uniform crossover in our experiments.

- **Mutation operator:** In IEEE floating-point format, it is found that some bits are less efficient during bit mutation. The sign bit mutation is useless as the estimated parameter are a positive real numbers. Similarly, if we mutate at a very high exponential bit or at a very low fractional bit, the whole string will respectively be $2^{\pm 128}$ times the original or only be changed slightly. In fact, these mutations may be too severe or negligible. Depending on Sensitivity analysis on different bit mutations, a weighted bit mutation is provided.
- **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 10000 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
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

5. SPRT ANALYSIS OF DATA SETS : TIME DOMAIN

In this section, the developed SPRT methodology is shown for a software failure data which is of time domain.

In this section the decision rules based on the considered mean value function for five different data sets, borrowed from Pham (2006), Xie *et al*., (2002) are evaluated. Based on the estimates of the parameter 'b' in each mean value function, we have chosen the specifications of $b_0 = b - \delta$, $b_1 = b + \delta$ equidistant on either side of estimate of b obtained through a data set to apply SPRT such that $b_0 < b < b_1$. Assuming the value of $\delta = 0.002$, the choices are given in the following table.

Table 5.1: Estimates of a, b & Specifications of b_0, b_1 for Time domain

Data Set	Estimate of 'a'	Estimate of 'b'	b_0	b_1
1	75.028818	0.046545	0.044545	0.048545
2	95.382252	0.075794	0.073794	0.077794
3	99.447446	0.019155	0.017155	0.021155
4	98.044273	0.107671	0.105671	0.109671
5	86.303732	0.119942	0.117942	0.121942

Using the selected b_0, b_1 and subsequently the $m_0(t), m_1(t)$ for the model, we calculated the decision rules given by Equations 3.4 and 3.5, sequentially at each 't' of the data sets taking the strength (α, β) as (0.05, 0.3). These are presented for the model in Table 5.2.

Table 5.2: SPRT analysis for 5 data sets of Time domain data

Data Set	N(t)	Acceptance region (≤)	Rejection Region (≥)	Decision
1	1	10.8987447	132.732408	*Accept*
2	1	15.842602	118.829646	*Accept*
3	1	11.207816	31.140751	*Accept*
4	1	1.483368	140.712095	*Accept*
5	1	-2.58E+02	8.66E+02	*Continue*
	2	-3.05E+02	9.76E+02	
	3	-3.35E+03	7.94E+03	
	4	-3.85E+03	9.11E+03	
	5	-5.51E+03	1.29E+04	
	6	-2.22E+04	5.11E+04	
	7	-3.84E+04	8.83E+04	
	8	-5.80E+04	1.33E+05	
	9	-5.71E+05	1.31E+06	
	10	-2.39E+06	5.48E+06	
	11	-2.52E+06	5.78E+06	
	12	-5.05E+06	1.16E+07	
	13	-7.37E+06	1.69E+07	
	14	-2.05E+07	4.68E+07	
	15	-2.61E+07	5.98E+07	
	16	-1.51E+08	3.45E+08	
	17	-2.80E+09	6.41E+09	
	18	-3.85E+09	8.82E+09	
	19	-4.80E+09	1.10E+10	

	20	-7.69E+09	1.76E+10	
	21	-2.56E+13	5.85E+13	
	22	-1.85E+14	4.24E+14	
	23	-2.88E+14	6.60E+14	

From the above table it is observed that a decision of either to accept or reject the system is reached well in advance of the last time instant of the data.

6. CONCLUSION

The table 5.2 of Time domain data as exemplified for 5 Data Sets shows that Exponential model is performing well in arriving at a decision. Out of 5 Data Sets of Time domain the procedure applied on the model has given a decision of acceptance for 4 and continue for 1 at various time instant of the data as follows. Data Set #1, #2, #3 and #4 are accepted at 1st instant of time. Data Set #5 is continued. Therefore, by applying SPRT on data sets it can be concluded that we can come to an early conclusion of reliable or unreliable software.