# PARAMETER ESTIMATION OF GOEL-OKUMOTO MODEL BY COMPARING ACO WITH MLE METHOD

**G.Lavanya[1], K.Neeraja[2], Sk.Ahamad Basha[3] , Dr.Y.Sangeetha[4]**

*[1]G.Lavanya, Dept. of Information Technology, Velagapudi Ramakrishna Siddhartha Engineering College, Andhra Pradesh, India*

*[2]K.Neeraja, Dept. of Information Technology, Velagapudi Ramakrishna Siddhartha Engineering College, Andhra Pradesh, India*

*[3]Sk.Ahamad Basha, Dept. of Information Technology, Velagapudi Ramakrishna Siddhartha Engineering College, Andhra Pradesh, India*

*[4]Dr.Y.Sangeetha, Dept. of Information Technology, Velagapudi Ramakrishna Siddhartha Engineering College, Andhra Pradesh, India*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract -** *Statistical Process Control (SPC) is the best choice to monitor software reliability process. It assists the software development team to identify and actions to be taken during software failure process and hence, assures better software reliability. In this project we propose a control mechanism based on the cumulative observations of failures which is ungrouped data by using an infinite failure mean value function G-O model, which is Non-Homogenous Poisson Process (NHPP) based. By comparing Ant Colony Method (ACO) with Maximum likelihood (MLE) approaches which is used to estimate the unknown parameters of the model and to identify the best approach among these in order to find out the failures at an early stages.*

***Key Words*: *Statistical Process Control (SPC), Ant Colony Method (ACO), Non Homogenous Poisson Process (NHPP), Maximum likelihood (MLE)***

## 1. INTRODUCTION

The monitoring of Software reliability process is a far from simple activity. In recent years several authors have recommended the use of SPC for software process monitoring. The main thrust of the paper is to formalize and present an array of guidelines in a disciplined process with a view to helping the practitioner in putting SPC to correct use during software process monitoring. Software is a program that provides instructions to processor to function, generating the desired result. Software is broadly classified as operating system and application software. The operating system carries out the basic operations of a processor while application software works on a level higher than operating system providing special services. Software plays a key role in the modern life. The life of a software system is considered in a probabilistic way in order to develop a quality measure for the system called software Reliability. The basic requirement is a specific probability model from statistical science to be modeled for software failure data in order to predict the future failure time of the system. In this project the software failure pattern is approximated by a Ant Colony Optimization method and maximum likelihood estimation method. The failure process is mathematically evaluated and empirically examined for live software failure data with the theory of mathematical statistics. Software reliability modelling and prediction during product development is an area of reliability that is getting more focus from software developers. The use of software reliability growth models plays an important role in measuring improvements, achieving effective and efficient test scheduling during the course of a software development project, determining when to release a product or estimating the number of service releases required to reach a reliability goal. Reliability is a real-world phenomenon with many associated real-time problems. One of the common practices in manufacturing industry is Statistical Process Control (SPC). The investigation on quantitative mechanisms as an aid to control process variation led to the application of the SPC since 1930's. The idea of applying SPC to software development however, is exemplified mainly by Capability Maturity Model (CMM) in mid 90's. Although its benefits are accepted for manufacturing companies, there have been many debates about its application in software development.

### 1.1 Software Reliability

Software Reliability is an important quality characteristic of a software which can evaluate and predict the operational quality of software system during its development. Software Reliability is the probability of failure free operation of software in a specified environment for a specified period of time Statistical Process Control (SPC) concepts and methods are used for improving the software reliability by identifying and eliminating the human errors in the software development process. In general Software has two user requirements namely reliability and availability. Reliability is required when the products non-performance will have the greatest impact, while availability is required when downtime will have the greatest impact. It is quite difficult to define the reliability and the best way is to measure the software reliability probabilistically, because we just can't say that reliability is 100% if the program is correct and 0% if the program is incorrect. Reliability of software is not deterministic, as a faulty program can still give correct output sometimes. Measuring software reliability alone does not solve the problem of achieving reliable software; it merely reflects the quality of software on hand. Testing is usually a lengthy process in the software industry accounting for 40-50% of the development process. The bugs detected as time elapses is used to update the reliability information of the software. This information could be translated into determining the testing time or resources required in order to meet various criterions of reliability or cost. The reliability of the software is usually estimated by devising mathematical models describing a typical behavior of a debugging process.

## 1.2 Ant Colony Optimization

The basic idea of software reliability modeling is to predict the reliability of the software with its failure data. During the past 40 years, nearly one hundred software reliability growth models (SRGM) have been proposed. Two traditional methods in parameter estimation are maximum likelihood and least squares method. Ant colony algorithm is a brand-new bionic simulated evolutionary algorithm, which has been applied to many fields. It has a lot of virtues. First, it uses a positive feedback, parallel and self-catalyze mechanism. Thus it has an excellent robustness and is easy to collaborate with other methods. Second, it has a well performance to the optimization problem, and can get rid of traditional optimization methods' weakness. Third, coding and operation with ant colony algorithm is easy, and it has a good convergence rate. Therefore, this algorithm is suitable for the parameters estimation of software reliability models. Ant Colony Optimization method works in the Least Square's principle. Ant colony optimization is based on the technique known as Swarm Intelligence, which is a part of Artificial Intelligence.

## 1.3 Maximum Likelihood Estimation

Based on the time domain data specified in this chapter, the software failure process is demonstrated through Mean Value Control charts. To monitor the software reliability the cumulative values of the failure data is considered. The concept of taking cumulative quality values is very different as well as new approach and at the same time it is very helpful when assessing the reliability of a software process. The maximum likelihood estimation is used to estimate the unknown parameters a, b. The Maximum Likelihood Estimates (MLE's) of the parameters are computed using Newton Raphson iterative method for the given time domain cumulative data. The mean value m(t) can be computed once the parameters a, b are estimated.

## 1.4 Non Homogenous Poisson Process

The Non-Homogeneous Poisson Process (NHPP) based models are the most important models because of their simplicity, convenience and compatibility. The NHPP based software reliability growth models are proved quite successful in practical software reliability engineering. The NHPP model represents the number of failures experienced up to certain time. 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.

## 1.5 Problem Statement

The monitoring of Software reliability process is a far from simple activity. A failure is a departure of system behavior in execution from user requirements and it is the result of a fault. A fault is a defect that causes or can potentially cause the failure when executed. The objective is to probe the software for faults by subjecting the software to the same statistical distribution of inputs that is expected in operation so that these faults can be removed and its reliability thus is improved. Generally in software failures occur. Our aim is to find out the failure rate at early stages by considering ungrouped data which is time domain data. Rapid increasing of larger volumes of data and software in the internet day by day, there is a need for the people to have mechanism to assess the software for reliability.

**Justification:** Statistical Process Control (SPC) is the best choice to monitor software reliability process and applying Software reliability models. It assists the software development team to identify and actions to be taken during software failure process and hence, assures better software reliability.

**Objectives:**

- Rapid increasing of larger volumes of data and software in the internet day by day, there is a need for the people to have mechanism to assess the software for reliability.
- To find out the parameters by applying MLE Method
- The mean value of G-O model is to be evaluated.
- Statistical Process Control (SPC) is the best choice to monitor software reliability process which is applied on software reliability growth model.
- It is observed that we are able to come up with an early conclusion about the reliability or unreliability of a software product

- Early stages we detect the reliability and Compare two Methods and find out the best method to find the failure rate at early stages

## 2. PROPOSED SYSTEM

Statistical Process Control (SPC) is a quality control method with which we can monitor whether the process is operating at its full potential. SPC can be applied for any process to assure that the product is meeting the required specifications by measuring the output [2]. It is said to be a powerful tool that can be used to enhance the quality as well as the productivity of a process. Compared with other quality control methods like "inspection", the main advantage of using SPC is it emphasizes early detection and prevention of problems, rather than the correction of problems after they take place. SPC also identifies the bottlenecks, waiting times, and other sources of delays within the process during its manufacture. The proper use of Control Charts brings stability and predictability to key processes.

### 2.1 Assessing Software Quality using SPC

Statistical Process Control (SPC) is an important tool with which we can monitor the reliability of developed software. For monitoring the software reliability, software failure data is very much essential that is available in two different types. The time domain data records the time of the failure occurrences and interval domain data records the number of failures in a given time interval. In this paper, we want to monitor the reliability of a software using SPC based on order statistics for time domain data. The main intention of this paper is to give a systematic procedure to show how SPC can be used to monitor the software reliability[1]. SPC is a powerful tool not only for improving the productivity of manufacturing procedure and it can also be used for software development. SPC is a method of process management through application of statistical analysis, which involves and includes defining, measuring, controlling, and improving the software processes. This chapter mainly focuses on the use of SPC for assessing the software reliability with G-O Model which is based on Non Homogeneous Poisson Process (NHPP)[3]. The Parameters are estimated using Maximum Likelihood Estimation (MLE) and ACO.

### 2.2 Model Description

**G-O Model:** Parameter estimation plays a significant role in software reliability prediction. In this chapter it is explained how the unknown parameters are estimated using MLE and ACO technique for the specified reliability growth model[1]. The parameters are estimated from the time domain data and the mean value function of **G-O Model** is,

$$\boxed{m(t) = a\,(1-e^{-bt})}$$

**a**=estimate of the expected total number of failures to be eventually detected.

**b**=number of faults (in 1hr)

We find time **t**, unknown parameters **a** and **b**

**The Goel-Okumoto model is based on the following assumptions**:

1. All faults in a program are mutually independent from the failure detection point of view.

2. The number of failures detected at any time is proportional to the current number of faults in a program. This mean that the probability of the failures for faults actually occurring, i.e., detected, is constant.

3. The isolated faults are removed prior to future test occasions.

4. Each time a software failure occurs, the software error which caused it is immediately removed , and no new errors are introduced. This is shown in the following differential equation:

Where *a* is the expected total number of faults that exists in the software before testing and *b* is the failure detection rate or the failure intensity of a fault.

### 2.3 Parameter Estimation Based on MLE and ACO Method

In statistics, MLE is a method of estimating the parameters of a statistical model. When applied to a data set and given a statistical model, maximum-likelihood estimation provides estimates for the model's parameters. The method of maximum likelihood corresponds to many well-known estimation methods in statistics. MLE would accomplish the estimates by taking the mean and variance as parameters and finding particular parametric values that make the observed results the most probable. The Maximum likelihood estimation (MLE) is the most useful technique for deriving the point estimators[5]. Parameter estimation is of primary importance in software reliability prediction. Once the analytical solution for m (t) is known for a given model, parameter estimation is achieved by applying a technique of Maximum Likelihood Estimate (MLE).

- The failure data is collected in time domain data. The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability (likelihood) of the sample data.

- In reliability applications, data sets are typically small or moderate in size. For large, complete data sets, both the LSE method and the MLE method provide consistent results.
- Maximum likelihood estimation begins with writing a mathematical expression known as the ***Likelihood Function*** of the sample data.

$$\sum_{i=1}^{n}[y_i - y_{i-1}].log[m(t_i) - m(t_{i-1})] - m(t_n) = LogL = LogL$$

**Advantages of the MLE :** parameter estimates are more precise/accurate. The estimated variance is smaller. The calculations use more of the information in the data.

Ant Colony Optimization is an algorithm developed by Dorigo in 1994 inspired upon stigmergic communication to find the shortest path in a network. Typical examples are telephone, internet, and any problem that can be described as Travel Salesman Problem[4]. Used/adopted by British Telecom MCI Worldcom, Barilla, etc. Advantage of algorithm is that, as ants do, it allows dynamic rerouting through shortest path if one node is broken. Most other algorithms instead assume that the network is static.

It was discovered that while ants are searching for food, they secret pheromone which will be a guide for other ants to follow the route. This pheromone is volatile in nature, so the ants will follow the route as long as signal is strong. If no ant has chosen a path and a path has been selected long time back, then ants will find difficulty in searching for the food. The Ant Colony algorithm idea is used based on the above fact. It will allow us to find optimal solutions from a set of candidate solutions. Basic Ant Colony Algorithm discussion can be found in . Changes should be made to the ant colony algorithm which has been used originally in search of discrete best networks, to suit to the parameter estimation problem

$$\text{m (J)}=\sqrt{\sum_{t=0}^{T}[(m(t)-u(t))]\hat{}2}$$

**Advantage of Aco:** Inherent parallelism. Positive Feedback accounts for rapid discovery of good solutions..

## 2.4 Algorithm for ACO

**Algorithm 1: Parameters estimation method based on ant colony algorithm**
Input: m(*t*) , *m* (*t*) :
Output: fitting result
Comment: m(*t*) :software reliability growth model
*m*(*t*): actual failure number

```
begin
Set the initial value of P0, r, Nmax and m;
Divide the solution space into n1 x n2 areas and Put the ants in these areas randomly;
Calculate initial fitness tk =Jk for each ant k as (1)
N¬0 ;
while N < Nmax
k¬0 ;
while k< m
Calculate travel probability pk for each ant k as (2);
if pk < P0
else
Ant k searches in its current' area locally;
Calculate its new fitness Jk as (1);
Else
Ant k searches globally;
Calculate its new fitness J1
k as (1);
end if
if J 'k < Jk
Update the ant's position;
else
The ant holds still;
end if
Calculate the ant's pheromone t k as (3);
K¬ k+1 ;
end while
Record the best value and the ants' position;
N ¬N + 1
end while
Output the position of the ant which has the best value, namely the fit result;
```

## 2.5 Mathematical Derivation for MLE and ACO

The parameters are estimated for the time domain data. Once the analytical solution for m (t) is known for a given model, parameter estimation is achieved by applying a well known technique Maximum Likelihood Estimation (MLE) and ACO.

The constants 'a', 'b' and 'c' which appear in the mean value function and various other expressions are called the parameters of the model [4]. In order to assess the software reliability, the parameters a, b and c are to be known or they are to be estimated from software failure data. The experiments are conducted to obtain 'n' independent observations, $t_1$, $t_2$..., $t_n$. The likelihood function for time domain data is given by

$$L = \prod_{i=1}^{N} abe^{-(bt)}$$

Taking natural logarithm on both sides,

$$\text{Log L}=\sum_{i=1}^{n}(abe^{-(bt_i)})\text{-a}[1\text{-}e^{-(bt_n)}]$$

The log likelihood equation is used to estimate the unknown parameters ‘a’, ‘b’. The parameters ‘a’, ‘b’ would be the solutions of the equations

Taking the partial derivative with respect to ‘a’ and equating to ‘0’

$$\frac{\partial LogL}{\partial a} = 0,$$

$$\boxed{a = \frac{n}{[1-(e^{-(bt_n)})]}}$$

Taking the partial derivative with respect to ‘b’ and equating to ‘0’

$$\frac{\partial LogL}{\partial b} = g(b) = 0$$

$$\boxed{g(b) = \sum_{i=1}^{n} t_i - \frac{n}{b} + nt_n \frac{(e^{-(bt_n)})}{1-(e^{-(bt_n)})} = 0}$$

Taking the partial derivative again with respect to ‘b’ and equating to ‘0’

$$\frac{\partial^2 LogL}{\partial b^2} = g'(b) = 0$$

$$\boxed{g'(b) = \frac{n}{b^2} - nt_n{}^2 \left\{ \frac{1}{1-(e^{-(bt_n)})} + \frac{(e^{-(bt_n)})}{1-(e^{-(bt_n)2}} \right\} (e^{(bt_n)})}$$

The parameter ‘b’ is estimated by iterative **Newton Raphson Method** using

$$\boxed{b_{n+1} = b_n - \frac{g(b)}{g'(b)}}$$

The values of ‘b’ in the above specified equations can be obtained using Newton Raphson Method. Solving the above equations simultaneously, yields the point estimates of the parameters a, b . These equations are to be solved iteratively and their solutions in turn when substituted in the log likelihood equation of ‘a’ would give analytical solution for the MLE of ‘a’. The values of b are obtained by applying numerical methods.

**ACO:** Parameter estimation plays a significant role in software reliability prediction [5]. In this chapter it is explained how the unknown parameters are estimated using ACO technique for the specified reliability growth model.

The parameters are estimated from the time domain data and the mean value function of G-O Model is,

$$\mathbf{m\ (t) = a\ [1\text{-}e^{-bt}]}$$

Let {N(t), t >0 } be the counting process, mean value function and intensity function of a software failure phenomenon. The mean value function m(t) is finite valued, non decreasing, non negative and bounded with the boundary conditions.

$$\mathbf{m(t)} = \begin{cases} 0, t = 0 \\ a, t = \infty \end{cases}$$

Here ‘a’ represents the expected number of software failures eventually detected. If λ (t) is the corresponding intensity function.. λ (t) is a decreasing function of *m* (*t*) as a result of repair action following early failures. A relation between *m*(*t*) and λ(t) is given by,

$$\lambda(t) = \frac{b}{2a}[a^2 - m^2(t)]$$

Where ‘b ‘is a positive constant, serving the purpose of constant of proportional fall in λ(t). This relation indicates a decreasing trend for λ (t) with increase in *m*(*t*) From the fact that λ(t) is the derivative of *m* (*t*) we get the following differential equation, whose solution is and an NHPP with its mean value function and its intensity function is,

$$\mathbf{m\ (t) = a\ (1\text{-}e^{-bt})\ \text{-}(1)}$$

Differentiating w.r.to‘t’

$$\lambda(t) = \frac{2abe^{-bt}}{(1+e^{-bt})^2} \quad \text{-----------} \quad (2)$$

Suppose we have ‘n’ time instants at which the first, second, third..., nth failures of Software are experienced. In other words if *Sk* is the total time to the k th failure, *Sk* is an observation of random variable *Sk* and ‘n’ such failures are successively recorded. The joint probability of such failure time realizations S1, S2, S3......Sn is the constants ‘a‘, ‘b‘ which appear in the mean value function for G-O and hence in NHPP[3], in intensity function (error detection rate) and various other expressions are called parameters of the model. In order to have an assessment of the software reliability ‘a’, ‘b’ are to be known or they are to be estimated from a software failure data.

$$L=e^{-m(s_n)}.\prod_{k=0}^{n}\lambda(s_n) \quad \text{--------} \ (3)$$

The function given in above equation is called the MLE function of the given failure data. Values of ‘a’, ‘b’ that would maximize L are called MLE and the method is called MLE method of estimation. Accordingly ‘a’, ‘b’ would be solutions of the equations. Substitute equation’s (1) and (2) in equation (3) and taking logarithms, differentiating with respect to ‘a’, ’b’ and equating to zero.

Partial differentiate the above equation w.r.to ‘a’

$$\frac{\partial \log L}{\partial a} = 0$$

Then we have the expression for ‘a’

$$\boxed{a = n[\frac{1+e^{-bs_n}}{1-e^{-bs_n}}]}$$

Partial differentiate the above equation w.r.to ‘b’

$$\frac{\partial \log L}{\partial b} = 0$$

Then we get a function ‘g(b)’

$$\boxed{g(b) = \sum_{k=1}^{n} s_k - \frac{n}{b} - 2\sum_{k=1}^{n-1}\frac{s_{k.}e^{-bs_k}}{1+e^{-bs_n}} - \frac{2s_{n.}e^{-bs_n}}{1+e^{-bs_n}}\left[1-\frac{n}{1-e^{-bs_n}}\right] = 0}$$

Again partial differentiate the above equation w.r.to ‘b’

$$\frac{\partial^2 \log L}{\partial b^2} = 0$$

$$\boxed{g'(b) = \frac{n}{b^2} + 2\sum_{k=1}^{n-1}\frac{{s_k}^2 e^{-bs_k}}{(1+e^{-bs_k})^2}}$$

The value of ‘b’ can be obtained using Newton-Raphson method which when substituted in equation of ‘a’ gives value of ‘a’. Newton-Raphson method

## 2.6 ILLUSTRATIONS / TIME DOMAIN FAILURE DATA SET

The failure control charts are generated considering various failure software process and the procedure is illustrated with Time domain data set in this section.

**Table-1 Naval Tactical System (NTDS) Software Error (Pham Hong, 2005)**

NTDS Software Failure data set was extracted from information about failures in the development of software for the real time multi-computer complex of the US Naval Fleet Computer Programming Centre of the US Naval Tactical Data Systems (NTDS) (Goel 1979a). The software consists of 38 different project modules. The time horizon is divided into four phases: Production phase, test phase, user phase, and subsequent test phase. It was observed that 26 failures were found during the production phase, five during the test phase and the last failure was found on 4 January 1971. One failure was observed during the user phase, in September 1971, and two failures during the test phase in 1971. The table describes 26 software failures that are formed during the production phase.

| No. of Error | Inter-failure time | Cumulative failure time | No. of Error | Inter-failure time | Cumulative failure time |
|---|---|---|---|---|---|
| 1 | 9 | 9 | 14 | 9 | 87 |
| 2 | 12 | 21 | 15 | 4 | 91 |
| 3 | 11 | 32 | 16 | 1 | 92 |
| 4 | 4 | 36 | 17 | 3 | 95 |
| 5 | 7 | 43 | 18 | 3 | 98 |
| 6 | 2 | 45 | 19 | 6 | 104 |
| 7 | 5 | 50 | 20 | 1 | 105 |
| 8 | 8 | 58 | 21 | 11 | 116 |
| 9 | 5 | 63 | 22 | 33 | 149 |
| 10 | 7 | 70 | 23 | 7 | 156 |
| 11 | 1 | 71 | 24 | 91 | 247 |
| 12 | 6 | 77 | 25 | 2 | 249 |
| 13 | 1 | 78 | 26 | 1 | 250 |

## 2.7 FLOWCHART

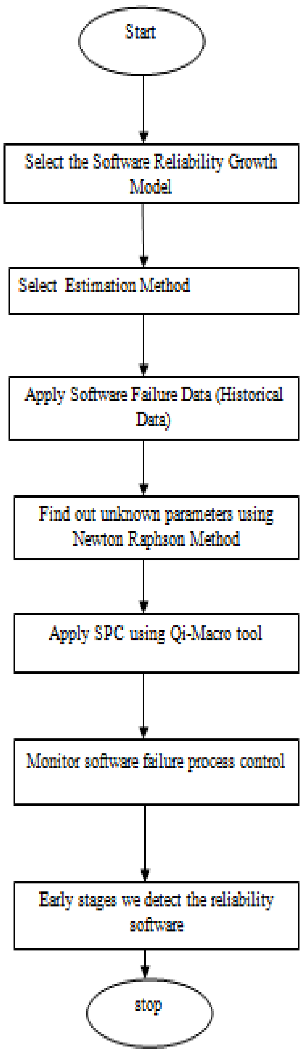

Start
Select the Software Reliability Growth Model
Select Estimation Method
Apply Software Failure Data (Historical Data)
Find out unknown parameters using Newton Raphson Method
Apply SPC using Qi-Macro tool
Monitor software failure process control
Early stages we detect the reliability software
stop

**Fig-1 : Flow Chart**

## 2.8 PARAMETER ESTIMATIONS

The parameters a and b are computed for the data sets specified in table 3 using well known Newton Raphson method. The values of ' b' obtained with Newton Raphson method are substituted in the equation to obtain the value of 'a'. For a software system the failures are random and can take place both during design phase or analysis phase and in some cases due to insufficient testing of software[1]. In this chapter the G-O model has been applied to the time between failures data. The distribution uses cumulative time between failure data for reliability monitoring.

**Table-2 Parameter Estimation of MLE and ACO**

| Failure data sets | No.of parameters | Estimated parameters | |
|---|---|---|---|
| | | a | B |
| **Table-1** | 26 | 55.01871 | 0.998899 |
| **Table-1** | 26 | 54.01781 | 0.9888 |

## 2.9 CONTROL LIMITS FOR ACO AND MLE

The software failure process is demonstrated through Mean Value Control charts for the time domain failure data set specified in Table 1. For monitoring the reliability of software process, cumulative failures have been taken into consideration. The maximum Likelihood Estimates a, b G-O Model are computed for the cumulative failure data count using Newton Raphson method which is a well-known iterative method. The mean value for each failure can be computed by substituting the values of 'a', 'b'

The mean value function of the G-O model is given by

$$\mathbf{m\ (t) = a\ (1-}e^{-bt}\mathbf{)}$$

Procedure for computing the control limits:

- Remove the term 'a' from the mean value function
- Equate the remaining function successively to 0.99865, 0.00135 & 0.5 and solve for 't', to get the 3 sigma corresponding control limits, upper control limit(UCL), Lower Control Limit(LCL) and Central limit(CL) of G-O model.

$$T_u = 0.99865$$

$$T_c = 0.5$$

$$T_L = 0.00135$$

Solving these equations the control limits are obtained and is described below

$$a\ (1-e^{-bt})=0.99865$$

$$\frac{a^b}{(t+a)^b} = 0.00135$$

$$(t+a)^b = \frac{a^b}{0.00135}$$

Apply log on both sides

= >b Log (t+a) = b Log a – Log 0.00135

$$=> \text{Log}(t+a) = \text{Log}\, a - \frac{1}{b}\text{Log}\, 0.00135$$

$$=> t + a = e^{\left\{Log\, a - \frac{1}{b} Log\, 0.00135\right\}}$$

$$t = \left[e^{\left\{\text{Log}\, a - \frac{1}{b}\text{Log}\, 0.00135\right\}} - a\right] = T_U$$

$$t = \left[e^{\left\{\text{Log}\, a - \frac{1}{b}\text{Log}\, 0.99865\right\}} - a\right] = T_L$$

$$t = \left[e^{\left\{\text{Log}\, a - \frac{1}{b}\text{Log}\, 0.5\right\}} - a\right] = T_C$$

The control limits in the mean value control chart indicates that the point above the m($t_u$) ((UCL)is an alarm signal. A point below the m($t_L$) (LCL) is an indication of better quality of software. The point within the control limits indicates that the process is stable.
The selection of proper SPC charts is essential for effective Statistical Process Control implementation and use. There are many types of charts and based on the data and situation the selection is to be made (Ronald.P.Anjad, 1995). Variable control charts are designed to control product or process parameters and X-bar and R charts are of that type. Attributes are characteristics of a process which are stated in terms of good or bad, accept or reject, etc. Attribute charts are not sensitive to variation in the process as variable charts. For attribute data there are p-charts, c-charts, np-charts and u-charts. The control charts are named as Failure Control Chart with which we can assess the software failure phenomena on the basis of given inter-failure time data. The control limits of the failure data set for Table 1

**Table-3 Parameter Estimates and Control Limits for Time Domain data for MLE**

| **Data sets** | **No of failures** | **Estimated Parameters** | | **Control Limits** | | |
|---|---|---|---|---|---|---|
| Table -1 | 26 | a | b | UCL | CL | LCL |
| | | 55.01871 | 0.998899 | 54.9444347 | 27.509 | 0.07427 |

**ACO:** The software failure process is demonstrated through Mean Value Control charts for the time domain failure data sets in table. For monitoring the reliability of software process, cumulative failures have been taken into consideration. The ACO Method G-O Model of are computed for the cumulative failure data count using Newton Raphson method which is a well-known iterative method. The mean value for each failure can be computed by substituting the values of a, b . The mean value function of the G-O Model

**m (t) = a (1-e-bt)**

**Procedure for computing the Control Limits:**

Remove the term 'a' from the mean value function. Equate the remaining function successively to 0.99865, 0.00135 & 0.5 and solve for 't', to get the 3 sigma corresponding control limits, upper control limit(UCL), Lower Control Limit(LCL) and Central limit(CL) of G-O Model. It gives

$$t = \frac{7.300122639}{b} = TU$$

Similarly,

$$t = \frac{0.002700002}{b} = TL$$

$$t = \frac{1.098612289}{b} = TC$$

The control limits are such that the point above the m($t_u$) (UCL) is an alarm signal. A point below the m($t_l$) (LCL) is an indication of better quality of software. A point within the control limits indicates stable process.

**Table-4 Parameter Estimates and Control Limits for Time Domain data for ACO**

| Data sets | No of failures | Estimated Parameters | | Control Limits | | |
|---|---|---|---|---|---|---|
| Table-1 | 26 | a | b | UCL | CL | LCL |
| | | 54.01781 | 0.9888 | 54.9443 | 27.5036 | 0.074255 |

## 2.10 DEVELOPING FAILURE CONTROL CHARTS

Given 'n' inter-failure data, the values of m(t) are computed for each failure. The values of $T_C$ , $T_U$ and $T_L$ are calculated and then by substituting in m(t) we get the corresponding control limits, UCL, LCL and CL. The successive differences of the m(t)'s are taken which gives n-1 values. The graph is generated by taking the inter-failure times on X-axis and the n-1 values of m(t)'s on Y-axis and three control lines parallel to X-axis at m($T_L$), m($T_C$), m($T_U$) respectively constitutes failure control chart. To assess the reliability of a software, the control chart has to be generated for the given inter-failure time data. The successive differences of mean values of the Table 1 are computed and shown in Table 5 and Table 6 for MLE and ACO

**Table-5 : Successive differences of mean value function m(t) of Table 1 for MLE**

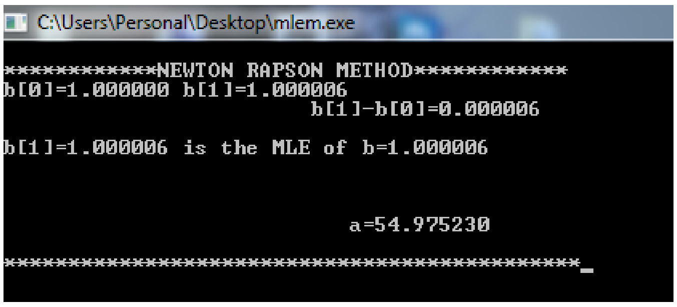
| S.No | Inter – Failure Time | Cumulative Failure Time | m(t) | Successive Differences of m(t) |
|---|---|---|---|---|
| **1** | 9 | 9 | 0.17198 | 2.680248 |
| **2** | 12 | 21 | 2.852228 | 2.436898 |
| **3** | 11 | 32 | 5.289126 | 1.859468 |
| **4** | 4 | 36 | 6.348594 | 0.294291 |
| **5** | 7 | 43 | 6.642885 | 0.294291 |
| **6** | 2 | 45 | 7.362949 | 0.720064 |
| **7** | 5 | 50 | 8.470581 | 1.107632 |
| **8** | 8 | 58 | 9.136521 | 0.66594 |
| **9** | 5 | 63 | 10.03676 | 0.90024 |
| **10** | 7 | 70 | 10.16243 | 0.125665 |
| **11** | 1 | 71 | 10.90158 | 0.739154 |
| **12** | 6 | 77 | 11.02235 | 0.120775 |
| **13** | 1 | 78 | 12.07962 | 1.057264 |
| **14** | 9 | 87 | 11.533 | 0.453377 |
| **15** | 4 | 91 | 12.64481 | 0.111816 |
| **16** | 1 | 92 | 12.97667 | 0.331858 |
| **17** | 3 | 95 | 13.30324 | 0.326574 |
| **18** | 3 | 98 | 13.94104 | 0.637793 |
| **19** | 6 | 104 | 14.0454 | 0.10436 |
| **20** | 1 | 105 | 15.15847 | 1.113071 |
| **21** | 11 | 116 | 18.15426 | 2.995794 |
| **22** | 33 | 149 | 18.73128 | 0.577016 |
| **23** | 7 | 156 | 24.83456 | 6.103281 |
| **24** | 91 | 247 | 24.94506 | 0.110506 |
| **25** | 2 | 249 | 25.00000 | 0.05494 |
| **26** | 1 | 250 | 26.00001 | |

**Table-6: Successive differences of mean value function m(t) of Table 1 for ACO**

| S.No | Inter – Failure Time | Cumulative Failure Time | m(t) | Successive Differences of m(t) |
|---|---|---|---|---|
| **1** | 9 | 9 | 0.17198 | 2.680248 |
| **2** | 12 | 21 | 2.852228 | 2.436898 |
| **3** | 11 | 32 | 5.289126 | 1.859468 |
| **4** | 4 | 36 | 6.348594 | 0.294291 |
| **5** | 7 | 43 | 6.642885 | 0.294291 |
| **6** | 2 | 45 | 7.362949 | 0.720064 |
| **7** | 5 | 50 | 8.470581 | 1.107632 |
| **8** | 8 | 58 | 9.136521 | 0.66594 |
| **9** | 5 | 63 | 10.03676 | 0.90024 |
| **10** | 7 | 70 | 10.16243 | 0.125665 |
| **11** | 1 | 71 | 10.90158 | 0.739154 |
| **12** | 6 | 77 | 11.02235 | 0.120775 |
| **13** | 1 | 78 | 12.07962 | 1.057264 |
| **14** | 9 | 87 | 11.533 | 0.453377 |
| **15** | 4 | 91 | 12.64481 | 0.111816 |
| **16** | 1 | 92 | 12.97667 | 0.331858 |
| **17** | 3 | 95 | 13.30324 | 0.326574 |
| **18** | 3 | 98 | 13.94104 | 0.637793 |
| **19** | 6 | 104 | 14.0454 | 0.10436 |
| **20** | 1 | 105 | 15.15847 | 1.113071 |
| **21** | 11 | 116 | 18.15426 | 2.995794 |
| **22** | 33 | 149 | 18.73128 | 0.577016 |
| **23** | 7 | 156 | 24.83456 | 6.103281 |
| **24** | 91 | 247 | 24.94506 | 0.110506 |
| **25** | 2 | 249 | 25.00000 | 0.05494 |
| **26** | 1 | 250 | 26.00001 | |

## 3. RESULT AND OBSERVATIONS

### 3.1 RESULT



```
C:\Users\Personal\Desktop\mlem.exe

*************NEWTON RAPSON METHOD************
b[0]=1.000000 b[1]=1.000006
                                        b[1]-b[0]=0.000006

b[1]=1.000006 is the MLE of b=1.000006


                                             a=54.975230

************************************************_
```

Fig-2 Result

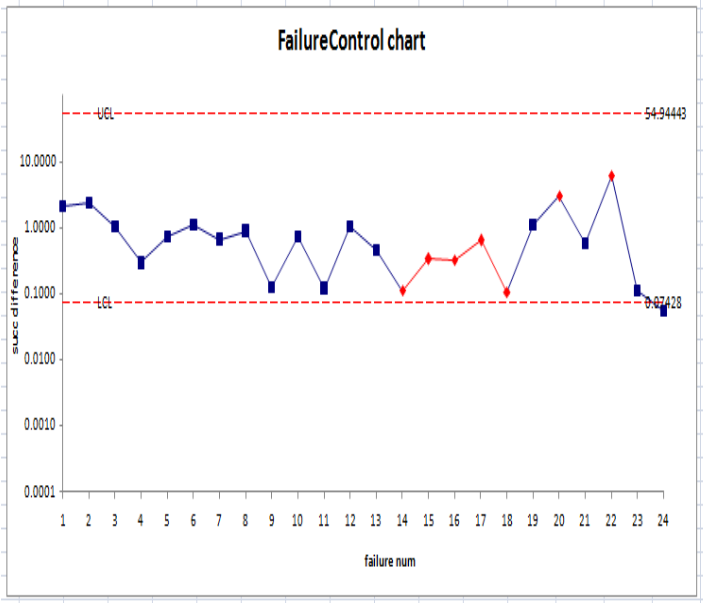


**Fig-3 Control Failure Chart for failure data Set for MLE**

The values of m(t) at Tc, Tu, Tl and at a given 26 inter failure times are calculated. Then m(t)'s are taken which leads to 25 values . The graph with the said inter failure times 1 to 25 on x-axis , the 26 values of m(t)'s on y- axis, and the three control lines parallel to x-axis at m(Tl), m(Tu), m(Tc) respectively constitutes failures control chart to assess the software failure phenomenon on the basis of the given time between failures data. This analysis data shows out of control signals i.e., below the LCL[3]. We Conclude that our method of estimation and the control chart are giving a +ve recommendation for their use in finding out preferable .By observing the Mean Value control chart we have identified that the failure situation is detected at 24th point.

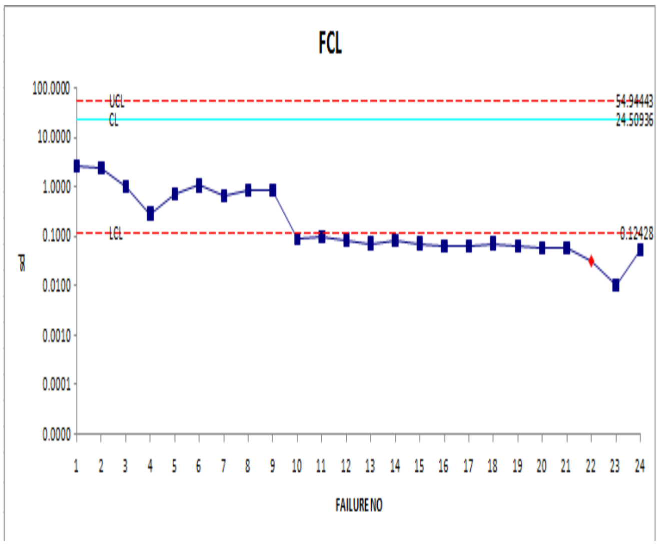


**Fig-4: Control Failure Chart for failure data Set for ACO**

By observing the Mean Value control chart we have identified that the failure situation is detected at 10th point.

### 3.2 OBSERVATION & DATA ANALYSIS

We present the analysis of two software failure data sets . The set of software errors analyzed is borrowed from a real software development project as published in Pham(2005), which in turn referred to (Pham (2005))as zhang et al.., (2000). There are advanced charts that provide more effective analysis. The basic types of advanced charts, depending upon the type of data are variable and attribute charts. Variable control charts are designed to control product or process parameters which are measured on a continuous scale. X bar, R charts are variable control charts.

Attribute charts are not sensitive to variation in the process as variable charts. However when dealing with attributes and used properly, by incorporating a real time analysis, they can be effective improvement tools. For attribute data they are p charts, c charts, np charts and u charts. We have named the control charts as failure control charts. The control charts help to assess the failure of s/w on the basis of the given inter failure time data.

Given 'n' inter-failure data, the values of m(t) are computed for each failure. The values of $T_C$ , $T_U$ and $T_L$ are calculated and then by substituting in m(t) we get the corresponding control limits, UCL, LCL and CL .The successive differences of the m(t)'s are taken which gives n-1 values. The Failure control charts from the Figures have shown out of control signals i.e., below LCL. By observing Failure Control Charts, it is identified that failures are detected at early stages. The early detection of software failure will considerably improve the software reliability. Software process improvement includes monitoring software development practices and actively seeking ways to increase value, reduce errors, increase productivity, and enhance the developer's environment.

**Lower Control Limit**: A point falling below the control limit indicates an alarming signal.

**Upper Control Limit**: A point above the control limit indicates the better quality.

**Control Limit**: If the points are falling within the control limits indicates that the software process is in stable.

We have used Qi-Macros Tool to get this failure chart which turns Microsoft Excel into a Power **Tool** for Lean Six Sigma and SPC. It draws control charts, pareto charts, histograms, fishbone, and more. ... **QI Macros** works right in Excel — no need to move or copy data into another application.

Form both the failure graphs we find that while comparing both the estimation methods ACO as shown failure rate at very early stages at point 10 and were as MLE as shown failure rate at point 24. So, We find that ACO is the best estimation method than MLE

## 4. CONCLUSION

In this project, we present a control mechanism based on the cumulative quantity between observations of time domain failure data using mean value function of G-O Model, which is based on Non Homogenous Poisson Process (NHPP).Ant Colony Optimization(ACO) method and MLE method is used to derive the point estimators of a two-parameter G-O Model. To calculate the parameter values and control limits using Statistic Process Control, we considered G-O Model .The Failure control charts from the Figures have shown out of control signals i.e., below LCL. By observing Failure Control Charts, it is identified that failures are detected at early stages. The early detection of software failure will considerably improve the software reliability. Software process improvement includes monitoring software development practices and actively seeking ways to increase value, reduce errors, increase productivity, and enhance the developer's environment.

S/w reliability is an important quality measure that quantifies the operational measures of computer systems. This model is primarily useful in estimating and monitoring software reliability. Which is used as a measure of quality. Equations to obtain the MLE of parameters based on the time domain data are developed. This analysis shows the control limits i.e below LCL. We conclude that method of estimation are giving a +ve recommendation for their use in finding out preferable control process of control signal. By observing the mean value control chart we detected the failure obtained at 24 and 10 point on both methods. Form both the failure graphs we find that while comparing both the estimation methods ACO as shown failure rate at very early stages at point 10 and were as MLE as shown failure rate at point 24. So, We find that ACO is the best estimation method than MLE. Hence our model is valid and very convenient for reliability. Hence our proposed Mean Value Chart detects out of control situation. The early detection of software failure will improve the software reliability. The methodologies in this model is better than X chart. Hence we conclude that model and ACO method is best choice for an early detection of software failures than MLE

## REFERENCES

[1] K. U. Sargut, O. Demirors; Utilization of statistical process control (SPC) in emergent software organizations: Pitfallsand suggestions; Springer Science + Business media Inc. 2016

[2] Hossain, S.A. and Dahiya, R.C., "*Estimating the Parameter of a Non Homogeneous Poisson Process Model for Software Reliability*", IEEE Transaction on Reliability, 42, 604-612.Barlow R, Proschan F, (2012) Mathematical Theory of Reliability, Wiley, NewYork

[3] Malaiya, Y.K., Karunanithi, N. and Verma, P.(1992), "*Predictability of Software Reliability Models*", IEEE Transactions on Reliability, Vol.41, No.4., 539-546.

[4] Jevtić, A.; Melgar, I.; Andina, D. (2009). "Ant based edge linking algorithm". 35th Annual Conference of IEEE Industrial Electronics, 2009. IECON '09. pp. 3353–3358.

[5] Andersen, Erling B. (2014); "Asymptotic Properties of Conditional Maximum Likelihood Estimators", *Journal of the Royal Statistical Society* **B** 32, 283–301