Effectiveness of Particle Swarm Optimization for Early Software Reliability Prediction

Krati Bajpai¹, Dr. P. Padmaja² and P. Ashwini³

¹,²,³Department of Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad, Telangana, India

Abstract: Software testing is extremely expensive and laborious, and it has been estimated that about half of software development costs are allocated to testing. Test data generation is the process of producing a set of data for software testing, based on a given criterion. Particle Swarm Optimization (PSO) uses an iterative method to optimize a solution of the problem. Modeling the software testing process to obtain the predicted faults (failures) depends mainly on the relationship between execution time, and the failure count or accumulated faults. In this project, I have explored the preliminary idea of Particle Swarm Optimization (PSO) technique in solving the software reliability growth modeling problem. The proposed approach has been used to estimate the parameters of Goel-Okumoto, Musa-Okumoto, Delayed S-Shaped and Power reliability growth models. The estimated parameters further used for decision making, such as, remaining faults in the software, future testing time and time to market for software product.

Keywords: Software Testing, Particle Swarm Optimization (PSO), Software Reliability Growth Models, Software Reliability Prediction

1. Introduction

During software testing phase of Software Development Life Cycle (SDLC), attempts are made in the form of white box or black box testing is to find out as many faults as possible which can result in failure of software. The reliability of the software increases by fixing the faults.

The data collected, in the form of faults, through these testing make a fault database. The database consist of number of faults in a successive interval of time. These faults further used for estimation of current reliability and prediction of future development of the growth in reliability.

The objective of proposed approach is to find optimized parameters of software reliability growth models. This approach would be helpful in determining which model is to be used for a given software, developed under specified development environment.

The purpose of the research is to make a model, and validate it with the help of given fault databases and to demonstrate that it can accurately define past failures and predicts future failures.

The problem handled during the research is to find out parameters of different reliability growth models, which can perform well in the given data set and able to predict faults in the near future in the software. If these parameters are of accurate value, then they will able to predict future faults accurately, which can further solve the problem of delivery to market and cost of the software.
2. Related Work

Y. Del Valle et al. [6] provides a comprehensive survey of Particle Swarm Optimization (PSO) on power system. It describes use of PSO on a highly nonlinear, non-stationary system with noise and uncertainties.

J. D. Musa and K. Okumoto [4] defined a new software reliability growth model that predicts expected faults. They had evaluated model efficiency using actual data and compared that with other existing reliability growth models.

John D. Musa [3] had given an approach of software reliability based on software execution time. Model developed by Musa estimates amount of time required for testing to achieve targeted reliability in advance. The execution time and faults count is taken in terms of mean time to failure (MTTF).

A. L. Goel [2] also proposed a software reliability growth model. He explained and provided a critical analysis of models and underlying assumptions, limitations and applicability of these models during software development life cycle.

3. Particle Swarm Optimization

Particle swarm optimization (PSO) is influenced by the organized nature of animal kingdom like bird flocking and fish schooling. PSO is an iterative method which gives stress on cooperation. Particle swarm optimization is a group based optimizing technique which is developed by Dr. Russell Eberhart and Dr. James Kennedy in the year 1995. Many of the researches have put their attention to PSO algorithm from its manifestation, as it is simple and effectual technique of optimization.

Particles in PSO can move in a multidimensional space to search an optimum solution. Every particle modifies its’ position according to its’ personal experience and according to the experience of its neighbor to get the best position [8], [9], [10].

In every iteration, each particle moves to a new location by updating its velocity by adding the three values given below:

- Its current velocity.
- A weighted random value in the direction of its’ personal best.
- A weighted random value in the direction of its neighborhood best.

The mathematical representation of above information can be referred to as velocity update equation and is given as:

\[ \mathbf{v} = \mathbf{v} + c_1 \cdot \mathbf{rand} \cdot (\mathbf{pBest} - \mathbf{p}) + c_2 \cdot \mathbf{rand} \cdot (\mathbf{gBest} - \mathbf{p}) \]  

Where

- \( \mathbf{p} \): particle’s position
- \( \mathbf{v} \): path direction
- \( c_1 \): weight of local information
- \( c_2 \): weight of global information
- \( \mathbf{pBest} \): best position of the particle
- \( \mathbf{gBest} \): best position of the swarm
- \( \mathbf{rand} \): random variable
The velocity update equation has three parts:

- The first part of the velocity update equation is known as “interia” or “momentum”. It concentrates on the direction in which the particle is currently moving.
• The second part of the velocity update equation is known as “self-knowledge” or “memory”. It concentrates on the previous best position obtained by a particular particle. In other words, it takes into account the personal best of a given particle.

• The third part of the velocity update equation is known as “cooperation” or “social knowledge”. It concentrates on the best position obtained by any particle in the given search hyperspace. In other words, it into account the global best.

Depending upon the new updated value of velocity, obtained from the velocity update equation, the position of the particles has been updated using the following position update equation:

$$ p = p + v $$

(2)

**Figure 3. Pseudo Code for PSO Algorithm**

4. Software Reliability Growth Models

Software reliability growth models use system data from testing phase of a software development life cycle to predict the software reliability. These models also help in predicting the remaining defects in a software [6], [7].

“Software reliability growth models are a statistical interpolation of defect detection data by mathematical functions. The functions are used to predict future failure rates or the number of residual defects in the code.”

[Alan Wood, Tandem Software Reliability Growth Models]

There is a parameter in many software reliability growth models associated with the total number of defects present in a software module. With the help of this parameter and the number of defects already known, we can calculate the remaining defects present in that software module.

In this paper, I have taken care of following four software reliability growth models:

4.1. Goel Okumoto Model

Goel – Okumoto model is a concave software reliability growth model. Goel - Okumoto model is one of the exponential non homogeneous Poisson Process (NHPP) model. It describes the occurrence of software failure during software testing.
The mean failure function of Goel–Okumoto model is given by following equation:
\[
\mu(t) = a(1 - e^{-bt}) \quad \text{where} \quad a \geq 0 \quad \text{and} \quad b > 0
\] (3)

‘a’ represents total number of expected defects in the code and ‘b’ represents the rate of failure detection.

4.2. Musa–Okumoto Model

Musa–Okumoto model is a concave software reliability growth model. This model is also a type of exponential non homogeneous Poisson Process model. It is the best model to predict the reliability of industrial datasets. The Musa–Okumoto model makes the assumption that there is an exponential decrease in failure intensity function with the observed number of failures.

The equation for mean failure function of Musa–Okumoto model is given by:
\[
\mu(t) = a \ln(1 + bt) \quad \text{where} \quad a \geq 0 \quad \text{and} \quad b > 0
\] (4)

‘a’ represents total number of expected defects in the code and ‘b’ represents the rate of failure detection.

4.3. Delayed S–Shaped Model

Delayed S–Shaped model is based on non-homogeneous Poisson Process. Delayed S–Shaped model shows the delay between detection of error and removal of that detected error. Delayed S–Shaped model is the S–Shaped software reliability growth model as the growth curve of observed cumulative number of defects in this model is S–Shaped.

Equation for mean failure function of Delayed S–Shaped model is given by:
\[
\mu(t) = a * (1 - (1 + bt) * e^{-bt}) \quad \text{where} \quad a \geq 0 \quad \text{and} \quad b > 0
\] (5)

‘a’ represents total number of expected defects in the code and ‘b’ represents the rate of failure detection.

4.4. Power Model

Power model is a non–exponential NHPP model. This model is also known as Duane’s Model. Power model is used to analyze the data of repairable systems. The mean value function of power model points to infinity very quickly. Because of this reason, this model has not received much recognition.

The equation for mean failure function of Power model is given by:
\[
\mu(t) = a*t^b \quad \text{where} \quad a \geq 0 \quad \text{and} \quad b > 0
\] (6)

‘a’ represents total number of expected defects in the code and ‘b’ represents the rate of failure detection.

5. Result Analysis of Data Set 1

The selected dataset is based on failure/fault count observed during testing. The small portion of dataset is given in Table 1 [1]. The dataset includes measured faults \(M_t\) and cumulative faults \(C_t\).

It is providing a statistical comparison between actual dataset and models chosen. Two parameters, a and b, are to be estimated from the fault count data for the models. The estimated values for the two parameters, for dataset given in Table 1 are

---

International Journal of Research

Volume 7, Issue XI, November/2018

ISSN NO:2236-6124

Page No:1449
Table 1. A Small Portion of Data Set 1

<table>
<thead>
<tr>
<th>Time</th>
<th>M_t</th>
<th>C_t</th>
<th>Time</th>
<th>M_t</th>
<th>C_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>4</td>
<td>4</td>
<td>102.</td>
<td>1</td>
<td>530</td>
</tr>
<tr>
<td>2.</td>
<td>0</td>
<td>4</td>
<td>103.</td>
<td>0</td>
<td>530</td>
</tr>
<tr>
<td>3.</td>
<td>7</td>
<td>11</td>
<td>104.</td>
<td>2</td>
<td>532</td>
</tr>
<tr>
<td>4.</td>
<td>10</td>
<td>21</td>
<td>105.</td>
<td>0</td>
<td>532</td>
</tr>
<tr>
<td>5.</td>
<td>13</td>
<td>34</td>
<td>106.</td>
<td>1</td>
<td>533</td>
</tr>
<tr>
<td>6.</td>
<td>8</td>
<td>42</td>
<td>107.</td>
<td>0</td>
<td>533</td>
</tr>
<tr>
<td>7.</td>
<td>13</td>
<td>55</td>
<td>108.</td>
<td>2</td>
<td>535</td>
</tr>
<tr>
<td>8.</td>
<td>4</td>
<td>59</td>
<td>109.</td>
<td>0</td>
<td>535</td>
</tr>
</tbody>
</table>

Figure 4. Parameter Values, Training RSME and Testing RSME

It is known that ‘a’ is an estimate of the expected total number of faults likely to be detected and ‘b’ is number of faults detected per unit time.

Figure 5. Result of Particle Initialization and Calculation
The result of Predicted Faults vs Expected faults is shown below:

![Graph of Predicted Vs Expected Faults](image)

**Figure 6. Predicted Faults Vs Expected Faults**

The result of root mean square error during supervised training is shown below:

![Graph of Training Root Mean Square Error](image)

**Figure 7. Root Mean Square Error during Training**

The combined result of the best location of particles for parameter ‘a’ of all the four implemented models is shown as:
Figure 8. Best Location of Particles for Parameter a

Figure 9. Best Location of Particles for Parameter b
6. Result Analysis of Data Set 2

The selected dataset 2 [1] is based on failure/fault count observed during testing. It is providing a statistical comparison between actual dataset and models chosen. Two parameters, a and b, are to be estimated from the fault count data for the models.
**Effectiveness of Particle Swarm Optimization for Early Software Reliability Prediction**

**Select Dataset**

<table>
<thead>
<tr>
<th>Select Model</th>
<th>Model Parameters</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goel Okumoto Model</td>
<td>(a = 567.151) (b = 0.0154159)</td>
<td>21.5515</td>
<td>36.2403</td>
</tr>
<tr>
<td>Musa Okumoto Model</td>
<td>(a = 388.19) (b = 0.0356267)</td>
<td>23.0747</td>
<td>44.1609</td>
</tr>
<tr>
<td>Delay S Shape Model</td>
<td>(a = 500.629) (b = 0.0653415)</td>
<td>15.4656</td>
<td>15.5672</td>
</tr>
<tr>
<td>Power Model</td>
<td>(a = 16.6416) (b = 0.809162)</td>
<td>28.1814</td>
<td>73.9821</td>
</tr>
</tbody>
</table>

*RMSE - Root Mean Square Error*

Figure 12. Parameter Values, Training RSME and Testing RSME

![Graph of Best Location of Particle for Parameter a](image)

Figure 13. Best Location of Particle for Parameter a
Figure 14. Best Location of Particle for Parameter b

Figure 15. Predicted Faults Vs Expected Faults
Figure 16. Training Root Mean Square Error

Figure 17. Front End – Consolidated Final Result for Dataset 1
7. Conclusion and Future Work

The models defined in previous chapter are software reliability growth models and used for reliability prediction and current reliability estimation as well as future faults in the software. In the proposed research all models are compared with two datasets and best model is selected for a particular database. It is known that one model is not best for every type of dataset and the same observation is noted down at the time of parameter finding of models and their comparison.

I am planning to explore and take advantage of other nature influenced optimization techniques such as Genetic Programming, Ant Colony Optimization etc. The specified techniques will be useful in developing a polynomial model structure. This model structure can provide a complex relationship between $\mu(t; \beta)$ and $\lambda(t; \beta)$ to better model the software reliability prediction process.

References