A Differential Evolution Approach for Software Testing Effort Allocation

Md. Nasar and Prashant Johri
School of Computing Science & Engineering, Galgotias University
Greater Noida, INDIA
Email: nasar31786@gmail.com, johri.prashant@gmail.com

Udayan Chanda
School of Business, Galgotias University
Greater Noida, INDIA
Email: udayanchanda@gmail.com

Abstract—Software reliability playing a major role in the industry. Because it carefully plan and guide to the developer and tester so that software developing team develop more reliable software faster and cheaper. Distribution of limited testing efforts to a software development project is a difficult task for team leaders. The challenges become complex when the nature of the development process is considered in the dynamic environment. For dynamic allocation of effort we proposed using differential evolution. Several software reliability growth models (SRGMs) have been proposed in last decade to minimize the total testing effort expenditures, but mostly under static assumption. The main intention of this article is to distribute total testing resource optimally under dynamic condition. An elaborate optimization policy based on the optimal control theory is proposed using differential evolution. Differential Evolution is an improved version of Genetic Algorithm for faster optimization.

Index Terms—Differential Evolution, testing effort allocation, optimal control theory

I. INTRODUCTION

Software development procedure is a complex process. It requires cautious planning and implementation to meet the objectives. Occasionally a developer must react speedily and aggressively to meet ever changing market demands. Maintaining software quality hinders fast paced software development, as numerous testing cycles are needed to ensure quality products. The challenge will become difficult when the development method is considered in the dynamic environment. To decrease uncertainty in the process often, organizations set up different project management tools to coordinate with the other components of the project. Software will be released to the users at the end of the testing phase of SDLC. With superior development and testing efforts, superior quality software can be guaranteed. But this will be time consuming and is undesirable in the prevalent competitive market conditions. Allocation of financial efforts to a software development project during the testing phase in the dynamic environment will be vital decision that a software manager has to make. During testing resources such as manpower and computer time is consumed. The fault detection and removal process will depend upon the nature and amount of resources used. Many software reliability growth models (SRGMs) is proposed in the last decade to discuss the minimization issue of the testing effort expenditures Chatterjee, Misra and Alam [4], Kapur and Garg [17]. Often these models are based on the assumptions that the testing effort consumption and testing time follows Rayleigh and exponential distribution. The time dependent behavior of the testing effort has been studied by many authors Basili and Zelkowitz [1], Kapur and Garg [17], Huang, Kuo, and Chen [9] and Yamada, Hishtani, and Osaki [27] Dohi et.al. [37] Proposed a neural network technique to approximate the optimal software release timing which minimizes the appropriate cost criterion via artificial neural networks. Huang [38] used a software cost model that is used to formulate whole software cost projects and discussed the optimal release policy based on reliability and cost, considering and efficiency testing effort.

Previous studies explored that exponential curve can be used if the testing resources are regularly consumed with respect to the testing time and Rayleigh curve otherwise. Logistic and Weibull-type functions were also used to describe the testing effort. Another school of thought assumes that the resource consumption can be expressed as an explicit function of the number of faults removed and calendar time Musa, Iannino, and Okumoto [20]. More recently, Tamura and Yamada [25] have proposed an SRGM based on stochastic differential equations in order to consider the active position of the open source project assuming that the software failure intensity depends on the time, and the software fault exposure phenomena on the bug-tracking system keep an unbalanced state. As discussed, over the three decades many SRGMs have been proposed to minimize the total expenditures, but mostly under static theory. Here in this paper we have tried to investigate an optimal resource
allocation plan of software during the testing phase under dynamic situation using DE (differential evolution). Differential Evolution (DE) algorithm is a branch of evolutionary computation developed by Rainer Storn and Kenneth Price [10, 12] discussed optimization problems in continuous domains. In DE, each variable’s value is represented by a real number. The main advantages of DE are its uncomplicated structure, simplicity of use, robustness and speed. DE is one of the best genetic type algorithms for solving problems with the real valued variables. Differential Evolution is a design tool of great utility that is immediately accessible for practical applications. It has been successfully applied to solve a wide range of optimization problems such as clustering [31], unsupervised image classification [32], digital filter design [33], optimization of nonlinear functions [34], chemical engineering processes [35] and multi-objective optimization [36]. Differential Evolution uses mutation as a search mechanism and selection to through the search near the prospective regions in the feasible region. Genetic Algorithms produce a series of populations by using selection mechanisms. GA use crossover and mutation as search mechanisms. The chief difference between Genetic Algorithms and Differential Evolution is that Genetic Algorithms rely on crossover, a mechanism of probabilistic and helpful exchange of information among solutions to locate superior solutions, while genetic algorithms use mutation as the primary search mechanism. Below figure shows the procedure of DE.

![Figure 1. Procedure of differential evolution.](image)

The rest of this paper is organized as follows. Section II gives the background of this paper. Section III describes the Model formulation and its solution. Section IV takes experiments with deferential evolution. Section V concludes the paper.

II. RESEARCH BACKGROUND

Software Reliability Growth Models (SRGMs) provide a measures to forecast future failure activities from known or assumed characteristics of the software, such as past failure data. Diverse types of SRGMs will use for different phases of the software development life cycle (SDLC). With the increasing demand to deliver good quality software, software development industries need to manage quality achievement and assessment. Some SRGMs have been proposed in software reliability literature under a different set of assumptions and testing environment. The proposed SRGM in this paper takes into account the time dependent variation in the testing effort. The testing efforts that manage the pace of testing for all the software projects are Musa et al. [20].

(a) Manpower which contains

- Failure detection professionals.
- Failure rectification professionals.

(b) Computer time.

Software testing, depending on the testing technique employed, can be implemented at any instance in the development process. Conventionally most of the test effort happens after the requirements will been defined and the coding process has been finished.

On a failure, the fault causing it is identified and then removed by failure rectification personnel. The computer facilities represent the computer time, which is necessary for failure detection and rectification. The influence of testing effort has also been included in some SRGMs; Kapur, Garg, and Kumar [18]; Kapur and Bardhan [16]; Kapur, Gupta, Shatnawi, and Yadavalli [19]; Myers [21]Myers [21]; Putnam [30]; Xie [26]; Ichimori, Yamada, and Nishiwaki [11]; Huang et al [9]; Pillai and Nair [23] planned that software should be developed and tested individually in a chronological step. Yamada et al. [27], Hou, Kuo, and Chang [7], Pham and Zhang [22] have recommended that system level software testing occurred only after the system was completely developed. Recently, Blackburn, Scudder, and Van Wassenhove [2], however, recommended that software development and system debugging and testing should be viewed as simultaneous activities. Kapur and Bardhan [16] investigated the association between the number of faults deleted with respect to time and/or testing effort. The authors planned that during the testing stage of a software development cycle, faults are removed in two phases: first a failure occurs and then the fault causing that failure is corrected; thus the testing effort will be spent on two separate processes; failure detection and failure rectification. In their paper, the authors developed an SRGM incorporating time delay not only between the two phases but also through the segregation of resources between them and proposed two alternate methods for controlling the testing effort for achieving the preferred reliability or error detection level. Chang [3] discussed the sequential software release policy based on a state space model. Chiang and Mookerjee [5] discussed a development process in which system integration occurs when the number of errors in the system will achieve a certain threshold. Jain and Gupta [14] have proposed optimal released policy for module based software. Jain and Priya [13], Zheng, S. [28] investigated software release policies to minimize development cost while satisfying a reliability objective in dynamic environment.

A. Nomenclature

- \(a\) is the initial fault content in the software.
- \(b\) is the fault detection rate.
- \(f(t)\) is the number of fault removed at time ‘t’.
- \(m(t)\) is the cumulative number of fault detected till time ‘t’ due to the testing effort \(w_1(t)\).
- \(T\) the planning period.
- \(C_1(m(t),w_2(t))\) Cost per unit at time ‘t’ for cumulative faults removed \(m(t)\) and debugging effort \(w_2(t)\).
- \(C_2\) is the cost of testing per unit testing efforts.
- \(W\) is the total resources utilized during the SDLC at any point of time ‘t’.

III. MODEL FORMULATION AND SOLUTION
We initiate our study by stating a common model with a very few assumptions. We are limiting our analysis to the case of a firm that controls its resources for testing and debugging under limited planning horizon. We are also consistent with the plan that the latent faults in the software system are discovered and removed during the testing period, and the number of faults remaining in the software system regularly decreases as the testing progresses. Thus, it is reasonable to assume the below differential equation:

\[ f(t) = \frac{d}{dt} m(t) = bw_1(t)(a - m(t)) \]  

(1)

Now assume the software organization wants to minimize the total expenditure over the limited planning horizon T. Then the objective function can be given by

\[ \text{Min} \int_0^T \left[ c_1(t)f(t) + c_2w_1(t) \right] dt \]

Subject to

\[ f(t) = \frac{d}{dt} m(t) = f(b, w_1(t)) \]

(2)

\[ \frac{m(t)}{a} \geq m_r \geq m(T) \geq m_a (= am_r) \]

where

\[ m(0)=0 \text{ and } w_1(t)+w_2(t)=W \]

\[ (w_1(t); w_2(t)) \geq 0 \text{ and } c_1(t) = c_1(m(t), w_2(t)). \]

The development period is [0, T] and the assumption m(T) \geq m_0 means that the firm aims to reaching at least the level m_0 at the end of the development period. The planning problem is to find the allocation of resources that minimizes the total expenditure.

To resolve the above optimization problem, let w_1(t) be an admissible control vector which transfers (m_0,t_0) to a target (m_0,T), where final state m(T) is specified but the final time T is not specified. Assuming that m*(t) is corresponding to w_1(t), then by Pontryagin Maximum principle, in order for \( w_1(t) \) to be optimal, it is necessary that there exists a non-zero, continuous vector function \( \lambda(t) \) and a constant scalar \( \lambda_0 \) such that (Sethi and Thompson [24]):

\[ W_1^*(t) \text{ maximizes} \]

\[ \left[-c_1(t) + \lambda(t)\right]f(t) - c_2w_1(t) \geq 0 \]  

(3)

We can interpret \( \lambda(t) \) as the marginal value of faults at time ‘t’, which should be negative because increasing the number of faults will increase the debugging cost. The physical interpretation of the Hamiltonian \( H \) can be given as follows: \( \lambda(t) \) stands for future cost incurred as one more fault is introduced in the system (at time t). Thus the Hamiltonian is the sum of testing cost \( c_2w_1(t) \), current cost \( c_1(t)f(t) \) and future cost \( \lambda(t)f(t) \). \( H \) represents the instantaneous total cost of the firm at time t.

The following necessary condition that hold for an optimal solution:

\[ \frac{\partial H}{\partial w_1} = 0 \Rightarrow -(c_1 - \lambda) f w_1 - c_2 = 0 \]

Other optimality condition is

\[ \frac{\partial^2 H}{\partial w_1^2} \leq 0 \Rightarrow -(c_1 - \lambda) f_{w_1w_1} - c_{1w_1w_1}f \]

\[ -2c_{1w_1}f_{w_1} \leq 0 \]

where \( c_{1w_1} = \frac{\partial c_1}{\partial w_1} \text{ and } c_{1w_1w_1} = \frac{\partial^2 c_1}{\partial w_1^2} \)

The above optimization problem is solved by a powerful computerized heuristic search and optimization method, i.e. DE Differential evolution (DE) Price and Storn, [29], [10],[12] has appeared as one of the most promising modern optimization technique in the past few years. A number of the reasons for the popularity of DE include easy implementation, fast convergence and little parameter tuning. DE typically relies on mutation as the search operator. Even though GA [15] can be measured as a better searching algorithm and many versions of GA were developed by numerous researchers, still it is considered as an algorithm with less convergence speed. To overcome this difficulty differential evolution (DE) was proposed, which can be measured as an improved version of GA and it has comparatively superior convergence speed.

Kenneth Price implemented the idea of using vector differences for perturbing the vector population, while solving Chebychev Polynomial fitting Problem, posed to him by Rainer Storn (Price and Storn, [29]). Since its inception, it has undergone a number of experimentations and computer simulations and finally established itself as a versatile and robust tool, as it is today. DE, which can be considered as a new member in the family of evolutionary algorithms, also uses operators like, crossover and mutation. Very similar to genetic algorithm, it also starts with an initial population of solutions but here all of them are represented by vectors. Here in this population based algorithm, all the “NP” randomly generated initial vectors are replaced with the better ones (generated by the specific variation operators)in every generation and this process repeats till the termination criteria is fulfilled.

The major step of the DE algorithm is given below:

Initialization
Evaluation
Repeat
- Mutation
- Recombination (crossover)
- Evaluation
- Selection
Until (termination criteria are met)

In the first step of the DE optimization process, the population of candidate solutions must be initialized. Typically, each decision parameter in every vector of the initial population is assigned to a randomly chosen value from within its corresponding viable bounds.
The Differential Evolution optimization process is carried out by applying the following three main genetic operations; mutation, recombination and selection. After the population is initialized, the operators of mutation, crossover and selection create the population of the next generation by using the current population.

However to solve the problem dynamically the initial value of λ(0) and f(0) must be given first then from objective function we will find the optimal value of effort at time t.

**TABLE I. PARAMETERS OF THE DE**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>200</td>
</tr>
<tr>
<td>Number of Generation</td>
<td>100</td>
</tr>
<tr>
<td>Crossover constant(CR)</td>
<td>0.7</td>
</tr>
<tr>
<td>Differentiation constant(F)</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Based on above parameters problem is solved using MATLAB version 7.4.0 [8], [6].

**IV. NUMERICAL ANALYSIS**

Assume that a=100, b=0.2, λ(0)=40, f(0)=2, c0=1000, c2=5000,

The parameter values of 'a' and 'b' can be estimated for any given dataset outside the normative model by considering Equation (1). Though these parameters are mutually correlated with w1(t) and w2(t), due to the interdependence between the two, one effort can be expressed in the form of other. In this analysis, the objective is to check the significance of allocation of the testing effort (w1), we use differential evolution to allocate testing effort.

Below figure shows the optimal allocation of testing effort.

![](image)

**TABLE II ALLOCATION OF EFFORT WITH RESPECT TO TIME.**

<table>
<thead>
<tr>
<th>Time(t)</th>
<th>Cumulative effort (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>358</td>
</tr>
<tr>
<td>3</td>
<td>504</td>
</tr>
<tr>
<td>4</td>
<td>539</td>
</tr>
<tr>
<td>5</td>
<td>546</td>
</tr>
<tr>
<td>6</td>
<td>598</td>
</tr>
<tr>
<td>7</td>
<td>600</td>
</tr>
</tbody>
</table>

**V. CONCLUSION**

In this paper we propose an alternate foundation for optimal allocation of testing resources. We used Differential evolution for this allocating resource. We suggest that software testing and debugging should be viewed as simultaneous behavior. During the investigation, we have examined to allocate effort in dynamic environment using differential evolution. This means that the developers and testers can devote their time and resources to complete their testing tasks based on well controlled expenditures.

**REFERENCES**


[8] Global Optimization Toolbox user’s Guide R2012a, the Mathworks INC


Dr. Prashant Johri working as a professor in School of computing science and Engineering, Galgotias University, Gr. Noida. He received his PhD degree in Marketing Models and Optimisation (Operational Research) from University of Delhi, Delhi. He has published numerous papers in the area of marketing models, optimisation, software reliability and inventory management in international journals and conference proceedings. He has also presented several research papers on marketing models and optimisation in national and international conferences. His current research interests include marketing models, inventory modeling in the context of innovation diffusion theory, software reliability growth modeling, dynamic optimisation techniques and quantitative techniques, etc. He is a Life Member of Society for Reliability Engineering, Quality and Operations Management (SREQOM)

Dr. Udayan Chanda is currently working as Assistant Professor in School of Business, Galgotias University, Gr. Noida, India. He received his PhD degree in Marketing Models and Optimisation (Operational Research) from University of Delhi, Delhi. He has published numerous papers in the area of marketing models, optimisation, software reliability and inventory management in international journals and conference proceedings. He has also presented several research papers on marketing models and optimisation in national and international conferences. His current research interests include marketing models, inventory modeling in the context of innovation diffusion theory, software reliability growth modeling, dynamic optimisation techniques and quantitative techniques, etc. He is a Life Member of Society for Reliability Engineering, Quality and Operations Management (SREQOM).