I.J. Modern Education and Computer Science, **2014, 11, 52-57**
Published Online November 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2014.11.07

Resource Allocation Policies for Fault Detection and Removal Process

Md. Nasar
School of Computing Science and Engineering, Galgotias University, Gr. Noida, India
Email: nasar31786@gmail.com

Prashant Johri and Udayan Chanda
School of Computing Science and Engineering, Galgotias University, Gr. Noida, India
Department of Management, Birla Institute of Technology & Science (BITS) Pilani. India
Email: {johri.prashant, udayanchanda}@gmil.com

Abstract—In software testing, fault detection and removal process is one of the key elements for quality assurance of the software. In the last three decades, several software reliability growth models were developed for detection and correction of faults. These models were developed under strictly static assumptions. The main goal of this article is to investigate an optimal resource allocation plan for fault detection and removal process of software to minimize cost during testing and operational phase under dynamic condition. For this we develop a mathematical model for fault detection and removal process and Pontryagain's Maximum principle is applied for solving the model. Genetic algorithm is used to find the optimal allocation of fault detection and removal process. Numerical example is also solved for resource allocation for fault detection and remoal process.

Index Terms—SRGM, Testing Effort Allocation, Correction-Removal Process, Optimal Control Theory, Genetic Algorithm.

I. Introduction

For the last few decades, it has been observed that computer systems have been widely used for problem solving, and there has been phenomenal increase in complexity of a system. A fault in the software system may produce huge loss in terms of money as well as time. There are numerous examples of failure of computer control system resulting in huge financial and other losses. Hence, it is necessary for an organization to invest resources in developing a software product that should be error free, reliable and also suitable for market conditions.

As industries are depending more and more on computer systems for day to day business, a reliable software system is needed by businesses to be efficient. Software testing comes under the software development life-cycle, and both are multistage processes, where each stage possesses a pre-specified activity or goal to deliver high quality products to a client. Software testing plays a very important role in software quality. Software testing interacts with every phase of software development life cycle, and the software reliability valuation is an important component to predict and evaluate the reliability of a software system. The model applicable to calculate the software reliability is called software reliability growth model (SRGM). The main goal of SRGM is to establish the relationship between fault observation and fault removal, and to calculate the reliability of the software. These reliability models have also been used for resource allocations.

Resources, such as manpower, CPU time and some hardware, are generally used during software testing process. Fault detection and correction process completely depends upon the nature of fault and volume of resources consumed for correction. Numerous SRGMs have been proposed to minimize the amount of testing effort consumed during testing but mostly under static condition. [1] first considered a simple software reliability model depending on the testing effort and formulated a testing resource allocation problem. [2] - [6] discussed the time-dependent behavior of testing efforts in their work. Generally, exponential curve is used to describe the performance of testing resources whenever spent equally; otherwise, Rayleigh curve is used. Weibull and Logistic type functions have also been used to describe testing efforts. [7] discussed optimization problem for testing resource allocations and for the software systems having modular structure and assumed that testing effort allocation depends upon the size and severity of the fault. Numerous SRGMs have been proposed in the last four decades in that mostly fault detection models were based on the assumption that detection and correction of faults is done concurrently. But, in reality there is always a time gap [8] - [12].

Generally, when a fault is detected, testing team reports it for correction, and then the development team rectifies the faults. Hence, evidently, there must be a time lag between fault detection and correction processes. [13] Determined the resource allocation problem by minimizing the mean number of remaining faults in the software modules with a budget constraint and a reliability aspiration. [14] Studied the optimal amount of resources desired for software module testing using the hyper-geometric software reliability growth model. [15]

Discussed the effort allocation in dynamic environment. The author used differential evolution for dynamic allocation of testing resources. Differential evolution is an improved version of genetic algorithm for faster optimization. The author also discussed a numerical example for allocating the testing resources.

[16] Discussed an optimal resource allocation problem for modular software systems throughout the testing phase. The main aim was to minimize the software development cost when the number of remaining faults is to be minimized, and a desired reliability objective is given. The authors analyzed the sensitivity of parameters of proposed software reliability growth models. In addition, the authors also present the impact on the resource allocation problem if some parameters are either overestimated or underestimated. The authors have evaluated the optimal resource allocation problems for various conditions by examining the behavior of the parameters with the most significant influence. For this purpose, a numerical example is also solved.

[17] Investigated dynamic programming approach for testing resource allocation problem when the software is developed in modules. For this, two optimization models are proposed for optimal allocation of testing resources among the modules of software. In the first model, authors maximize the total fault removal, subject to cost constraint. In the other model, other constraints representing aspiration levels for fault removals for each module of the software are added. The authors solved these models using dynamic programming technique. A dynamic programming approach for finding the optimal solution has also been proposed. The methods have been illustrated through numerical examples.

[18] Studied the association between the number of faults deleted with respect to testing effort and/or time. The authors assumed that throughout the testing stage of a software development life cycle (SDLC), faults are removed in two stages: first a failure occurs and then the fault causing that failure is corrected; hence the testing effort will be spent on two distinct processes; failure detection and failure alteration. In their paper, the authors developed a software reliability growth model 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 pre-specified reliability level or error detection level.

[19] Discussed a cost model for software, that is used to formulate whole software project cost and discussed the optimal release policy based on cost and reliability criteria.

[20] considered the cost factor in testing-resource allocation problems, but they only put the cost factor into the constraints, which means it is feasible if the total cost is less than budget. In fact, it is a profit for a company if the cost can be less than the budget provided that the customers' requirements are still satisfied. Thus he involved cost factor in the objective function together with reliability, which means that he is having multiple objectives in term of both maximizing system reliability and minimizing testing cost.

[21] investigated an optimal resource allocation plan to minimize the cost of software during the testing and operational phase under dynamic condition. An elaborate optimization policy based on the optimal control theory is used. The authors used learning curve phenomenon under dynamic environment for optimal allocation of testing resources. During the analysis, Author observed that due to the experience curve phenomenon, the effort required to fix an error keeps on decreasing with time. At the same time, testing effort keeps on increasing as in the later stages of a planning period it becomes hard to detect faults. For this purpose one numerical example is also illustrated. The model is based on the assumption that at any point of time the total resources allocated for debugging and testing is fixed.

The above literature review shows that most of the well-known SRGMs are built on static conditions but in real time, it is not true. In this paper, we have proposed mathematical optimization model that helps to assign allocation of resources for fault detection and fault correction under dynamic environment.

The paper is subdivided into the following sections. Section two and three describe the model for fault detection and correction processes, and the cost optimization modeling. In section four, we discuss the solution approach. Section five discusses the basic parameter for genetic algorithm. A numerical problem for fault detection and correction process Vs time using genetic algorithm is given in section six. Finally, in section seven, we conclude our paper with a discussion on results and findings.

Notations Used

T	:	The planning period.
'a'	:	Initial fault present in the software.
b_1	:	Fault detection rate.
b_2	:	Fault correction rate.
W	:	Total resources consumed during the software development at any point of time 't'.
$w_1(t)$	:	Resources consumed during the software development for testing purpose at any point of time 't'.
$w_2(t)$	:	Resources consumed during the software development to fix bugs at any point of time't'.
$f_d(t)$	:	Identified total number of faults till time *t*.
$f_r(t)$	:	Removed total number of faults till time *t*.
$c_1(f_d(t), w_3(t))$	:	Cost per unit at time 't' for cumulative fault detected is $f_d(t)$ with detection efforts $w_3(t)$.

$c_2\left(f_r(t), w_2(t)\right)$	:	Cost per unit at time ‘ t ’ for cumulative fault corrected is $f_r(t)$ debugging efforts $w_2(t)$
c_3	:	Cost of testing per unit effort at time ‘t’.

II. Model Development

Modeling Fault Detection and Correction Processes:

For perfect removal of faults, the predictable number of fault corrected is the same as predictable number of faults identified. However, in reality if a fault is identified, it is not necessary to correct the fault instantly. Hence, after identifying the fault, the debugging personnel ask the programmer to correct the fault. Thus, there must be a time gap between identification and removal processes. In general, the expected number of faults detected at time T is more than faults corrected at time T. Hence, the fault detection and removal processes are done in two stages. To begin with, let’s assume ‘a’ is the total fault content in the software. Therefore, it is necessary to assume the following equation of fault identification and removal process;

$$\begin{aligned} &x(t)=\frac{df_d(t)}{dt}= \quad b_1 w_1(t)(a-f_d(t)) \\ &\textit{and} \\ &y(t)=\frac{df_r(t)}{dt}=b_2 w_2(t)(\mathrm{f}_d(t)-f_r(t)) \\ &\textit{where} \\ &f_d(0)=0, f_r(0)=0 \end{aligned} \tag{1}$$

III. Cost Optimization Modelling

Now assume the software firm wants to minimize the total expenditure over the finite planning horizon. Therefore, mathematically the model can be given as;

$$\begin{aligned} &\min\int_0^T\left[c_1(t)x(t)+c_2(t)y(t)+c_3 w_1(t)\right]dt \\ &\textit{subject to} \\ &x(t)=\frac{df_d(t)}{dt}=b_1 w_1(t)(a-f_d(t)) \\ &y(t)=\frac{df_r(t)}{dt}=b_2 w_2(t)(\mathrm{f}_d(t)-f_r(t)) \\ &\textit{where} \\ &f_d(0)=0\ ,\mathrm{f}_r(0)=0\ , \\ &\mathrm{c}_1(t)=c_1\left(f_d(t), w_3(t)\right),\mathrm{c}_2(t) \\ &=c_2\left(f_r(t), w_2(t)\right)\ \textit{and} \\ &w_1(t)+w_2(t)+w_3(t)=w \\ &\left(w_1(t); w_2(t); w_3(t)\right)\geq 0 \end{aligned} \tag{2}$$

IV. Solution Procedure

To solve the problem for equation (2), Pontryagain’s Maximum principle is applied. The Hamiltonian function is as follows [22]:

$$\begin{aligned} &H(\mathrm{f}_d(t),\mathrm{f}_r(t),\lambda(t),w_1(t),w_2(t),w_3(t),t)\ = \\ &-\left[c_1(t)x(t)+c_2(t)y(t)+c_3 w_1(t)\right] \\ &+\lambda(t)x(t)+\mu(t)y(t) \end{aligned} \tag{3}$$

$\lambda(t)$ and $\mu(t)$ are the adjoint variables (shadow cost of and respectively), which satisfies the following differential equation.

$$\frac{d}{dt}\lambda(t)=\dot{\lambda}=-\frac{\partial H}{\partial f_d} \tag{4}$$

And;

$$\frac{d}{dt}\mu(t)=-\frac{\partial H}{\partial f_r} \tag{5}$$

Terminal condition for the differential equation (4) and (5) are given by $\lambda(T)=0$ and $\mu(T)=0$ respectively.

The adjoining variable $\lambda(t)$ represents per unit change in the objective function for a small change in $f_d(t)$ i.e. $\lambda(t)$ can be interpreted as marginal value of faults detected at time ‘ t ’. Similarly, $\mu(t)$ can be interpreted as marginal value of fault removed at time ‘t’. Thus, the Hamiltonian is the sum of current cost $(c_1 x+c_2 y)$ and the future cost $(\lambda x+\mu y)$. In short, H represents the instantaneous total cost of the firm at time’ t ’.

The following are the necessary conditions hold for an optimal solution:

$$\begin{aligned} &H_{w_1}=0\Rightarrow -c_{1w_1}(t)x(t)-(c_1(t) \\ &-\lambda(t))x_{w_1}(t)-c_3=0 \end{aligned} \tag{6}$$

$$\begin{aligned} &H_{w_2}=0\Rightarrow \\ &-c_{1w_2}(t)x(t)-c_{2w_2}(t)y(t)-(c_2(t) \\ &-\mu(t))y_{w_2}(t)=0 \end{aligned} \tag{7}$$

Where;

$$c_{1w_1}(t)=\frac{\partial c_1(t)}{\partial w_1(t)}$$
$$c_{1w_2}(t)=\frac{\partial c_1(t)}{\partial w_2(t)}$$
$$c_{2w_2}(t)=\frac{\partial c_2(t)}{\partial w_2(t)}$$

$$x_{w_1}(t) = \frac{\partial x(t)}{\partial w_1(t)}$$
$$y_{w_2}(t) = \frac{\partial y(t)}{\partial w_2(t)} \quad (8)$$

The other optimality conditions for Hamiltonian maximization are;

$$H_{w_1w_1} < 0 \quad and \; H_{w_1w_1}H_{w_2w_2} - H_{w_1w_2} > 0 \quad (9)$$

On solving equations (6) and (7), we get;

$$w_1^*(t) = -\frac{(c_1(t) - \lambda(t))b_1(a - f_d(t)) + c_3}{c_{1w_1}(t)b_1(a - f_d(t)} \quad (10)$$

And;

$$w_2^*(t) = -\frac{(c_2(t) - \mu(t))b_2(\mathrm{f}_d(t) - f_r(t)) + c_{1w_2}(t)x(t)}{c_{2w_2}(t)b_2(\mathrm{f}_d(t) - f_r(t))} \quad (11)$$

Using the assumption that the total resource is fixed i.e; $w_1(t) + w_2(t) + w_3(t) = w$. Thus,

$$w_3^* = w + w_1^* + w_2^* \quad (12)$$

Now, upon integrating equation (4) with the transversality condition, we have the future cost of detecting one more fault from the software;

$$\lambda(t) = -\int_t^T \left[\begin{array}{l} \frac{\delta c_1(t)}{\delta f_d} x(t) + (c_1(t) - \lambda(t)) \frac{\delta x}{\delta f_d} \\ + (c_2(t) - \mu(t)) \frac{\delta y}{\delta f_d} \end{array} \right] dt \quad (13)$$

Similarly, integrating equation (5) with the transversality condition, we have the future cost of removing one more fault from the software;

$$\mu(t) = -\int_t^T \left[\frac{\delta c_2}{\delta f_r} y(t) + \begin{pmatrix} c_2(t) \\ -\mu(t) \end{pmatrix} \frac{\delta y}{\delta f_r} \right] dt \quad (14)$$

Now taking time derivative of equation (6), we have;

$$\dot{w}_1 H_{w_1w_1} + \dot{w}_2 H_{w_1w_2} + x(t)H_{w_1f_d} + y(t)H_{w_1f_r} = -\dot{\lambda}\, x_{w_1} \quad (15)$$

Time derivative of equation (7) implies;

$$\dot{w}_1 H_{w_2w_1} + \dot{w}_2 H_{w_2w_2} + x(t)H_{w_2fd} + y(t)H_{w_2fr} = -\dot{\mu}\, y_{w_2} \quad (16)$$

Where;

$$H_{w_1w_1} = \frac{\partial^2 H(t)}{\partial^2 w_1(t)}$$
$$H_{w_1w_2} = \frac{\partial^2 H(t)}{\partial w_1(t)\partial w_2(t)}$$
$$H_{w_2w_2} = \frac{\partial^2 H(t)}{\partial^2 w_2(t)}$$
$$H_{w_if_r} = \frac{\partial^2 H(t)}{\partial w_i(t)\partial f_r(t)}$$
$$H_{w_if_d} = \frac{\partial^2 H(t)}{\partial w_i(t)\partial f_d(t)}, \text{ for } i = 1, 2.$$

To solve the above optimization problem, we used Genetic Algorithm (GA). Genetic Algorithm is a computerized search and heuristic optimization method for solving difficult type of problem which cannot be solved easily by general methods [23], [24] and [25].

V. Genetic Algorithm

Genetic algorithm is an optimization technique that mimics the process of natural selection. This heuristic approach is regularly used to generate solutions for optimization problems. Genetic algorithm belongs to evolutionary class of algorithm which is inspired by natural evolution, such as selection, crossover, mutation, and inheritance. Genetic algorithm is successfully applied in many fields such as bioinformatics, economics, physics, chemistry, computer science, mathematics and many others.

To solve the above problem using genetic algorithm, following steps will follow.

Chromosome Representation: Genetic algorithm starts with some initial population represented as chromosome. A chromosome consists of genes and each gene represents a specific solution for the problem.

Initial population: We will generate an initial population for total number of faults 'a' content in the software. It will take minimum and maximum values and it will generate initial population form this limit.

Fitness of a chromosome: Fitness of a chromosome quantifies the optimality of a solution (chromosome) so that particular solution may be ranked against all the other solutions.

Selection: Selection is the process of choosing two chromosomes from the whole population of the chromosomes. The chromosome with highest fitness value has high probability of selection. There are many methods to choose the best chromosome like tournament selection, rank selection, roulette wheel selection, Boltzman selection, steady state selection and some others.

Crossover: It is the process in which two chromosomes (strings) combine their genetic material (bits) to produce a new offspring which possesses both their characteristics. Two strings are randomly picked from the mating pool to cross over.

Mutation: By mutation individuals are randomly altered. These variations (mutation steps) are commonly very small. They will be applied to the variables of the individuals with a low probability (mutation rate or mutation probability). Normally, offspring will be mutated after being created by recombination.

The steps, selection, crossover and mutation, are then repeated till the stopping criteria are reached.

VI. Numerical Solution

Using GA approach, in this section, we discuss the numerical solution of the problem as discussed in section 3 in order to count the number of errors detected and corrected at time ‘t’. To solve this problem, we have used genetic algorithm and implemented through MATLAB 7.4.0 and C++. The parameters used to solve this problem are:

Total number of population Size: 200
Total number of generation: 100
Method of selection: Tournament selection
Rate of crossover: 0.8
Rate of mutation: 0.1

And the base values are as follows:

$a=100$ $b_1=0.3$ $b_2=0.3$
$w_1=0.40$ $w_2=0.42$ $w=1$ $\lambda(0)=100$
$\mu(0)=100$ $f_d(0)=0$ $f_r(0)=0$ $c_3=500$
$c_0=1000$ $b_0=1000$

The expected number of fault detected and corrected is as follows:

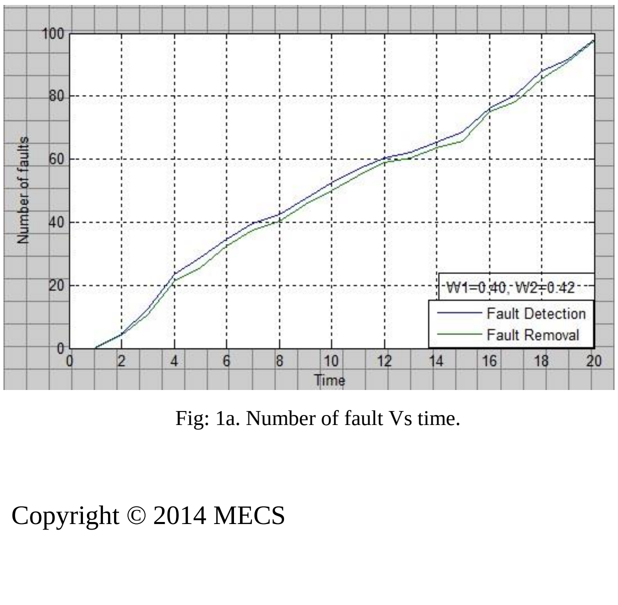

Fig: 1a. Number of fault Vs time.

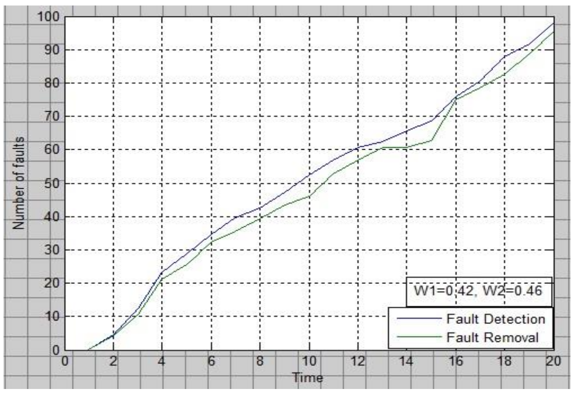

Fig: 1b. Number of fault Vs time.

Above fig. shows the fault detection and correction process Vs time. In first figure we use W1= 0.40 and W2=0.42, in second figure we use W1=0.42 and W2=0.46.

VII. Conclusions

In this paper, we propose an alternative foundation for optimal time allocation of fault detection and removal processes using genetic algorithm. We have considered fault detection and removal processes in two stages. During this study we have allocated fault detection and removal processes in dynamic environment. We describe a method to allocate time based on number of faults detected and corrected. This means that the tester and developer can dedicate their time and resources to finish off their testing and debugging tasks for well controlled expenditure. Simulation is done using genetic algorithm, MATLAB and C++ for the same.

References

[1] Ohtera, H. and Yamada, S.: Optimal allocation and control problems for software testing-resources. IEEE Transactions on Reliability, R-39 (2), 171-176 (1990).
[2] Putnam, L (1978), 'A general empirical solution to the macro software sizing and estimating problem', IEEE Transactions on Software Engineering, SE-4, 345-361.
[3] Yamada, S., J. Hishitani and S. Osaki (1993), 'Software Reliability Growth Model with Weibull testing effort: A model and application', IEEE Trans. On Reliability, R-42, 100-105.
[4] Basili, V.R. and M.V. Zelkowitz (1979), 'Analyzing medium scale software development', Proceedings of the 3rd International Conference on Software Engineering, 116-123.
[5] Kapur, P.K. and Garg, R.B. (1990), 'Cost Reliability optimum release policy for a software system with testing effort', OPSEARCH, 27, 109-116.
[6] Huang, C-Y, S-Y. Kuo and J.Y. Chen (1997), 'Analysis of a software reliability growth model with logistic testing effort function', Proceeding of 8th International Symposium on software reliability engineering, 378- 388.
[7] Kapur, P.K., Bardhan A.K.and Yadavalli V.S.S. (2007), 'On allocation of resources during testing phase of a modular software', International Journal of Systems Science, 38 (6), 493 - 499.

[8] Ohba, M. (1984),'Software reliability analysis models', IBM Journal of research and Development 28, 428-443.
[9] Schneidewind, N.F. (1975), 'Analysis of error processes in computer software', Sigplan Notices 10, 337-346.
[10] Xie, M. and Zhao, M. (1992), ' The Schneidewind software reliability model revisited.' Proceedings of the 3rd International Symposium on Software Reliability Engineering, 5, 184-192.
[11] Schneidewind, N.F. (1975), 'Analysis of error processes in computer software', Sigplan Notices 10, 337-346.
[12] Gokhale, S.S., Wong, W.E., Trivedi, K.S. and Horgan, J.R.,(1998), 'An analytic approach to architecture-based software reliability prediction' in Proceedings of the International Symposium on Performance and Dependability, September, pp. 13-22.
[13] Yamada S. Ichitmori T. Nishiwaki M., "Optimal allocation policies for testing-resource based on a software reliability growth model", Mathematical and Computer Modelling, 1995, 22(10-12), 295-301.
[14] Huo R.H., Kuo S.K., Chang Y.P., "Needed resources for software module test, using the hyper-geometric software reliability growth model", IEEE Trans. on Reliability, 1996, 45(4), 541-549.
[15] Md. Nasar, Prashant Johri and Udayan Chanda, "A Differential Evolution Approach for Software Testing Effort Allocation," Journal of Industrial and Intelligent Information, Vol. 1, No. 2, pp. 111-115, June 2013.
[16] C. Y. Huang, J. H. Lo, S. Y. Kuo and M. R, "Optimal Allocation of Testing-Resource Considering Cost, Reliability, and Testing-Effort," Dependable Computing, 2004. Proceedings. 10th IEEE Paci?c Rim International Symposium on 3-5 March 2004, pp. 103-112.
[17] Dohi, T., Nishio, Y., and Osaki, S. (1999), 'Optimal Software Release Scheduling Based on Artificial Neural Networks', Annals of Software Engineering, 8, 167-185.
[18] P. K. Kapur and A. K. Bardhan, “Testing effort control through software reliability growth modelling”, International Journal of Modelling and Simulation, vol. 22, pp. 90–96.2002.
[19] C.-Y. Huang, (2005), “Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency,” Journal of Systems and Software, vol. 77, pp. 139–155.
[20] Coit, D.W., Smith, A.E., 1996. Reliability optimization of series–parallel systems using a genetic algorithm. IEEE Trans. Reliab. 45 (2), 254–266.
[21] P.K. Kapur, Hoang Pham, Udayan Chanda & Vijay Kumar (2012): Optimal allocation of testing effort during testing and debugging phases: a control theoretic approach, International Journal of Systems Science, 2012, 1–12, iFirst, Taylor & Francis Publication.
[22] Sethi, S.P., and Thompson, G.L. (2005), Optimal Control Theory - Applications to Management Science and Economics (2nd ed.), New York: Springer.
[23] Goldberg, D. E., (1989), Genetic Algorithms: in Search Optimization and Machines Learning (New York: Addison-Wesley).
[24] David, L. Handbook of Genetic Algorithms. New York : Van Nostrand Reinhold. (1991).
[25] Deb K. (1995) Optimization for Engineering Design-Algorithms and Examples. Prentice Hall of India, New Delhi.

Authors' Profiles

Md. Nasar received his BCA degree from T. M. Bhagalpur University, Bhagalpur in 2002, Master in Computer Science from G. B. Pant University of Agriculture & Technology, Pantnagar, India in 2006. He has also received Microsoft Certified Technology Specialist (MCTS). At present, he is pursuing Ph.D in Computer Science from Galgotias University, Gr. Noida, INDIA. He is having 8 years of experience in Teaching, and Software Development. His research interest includes Software Reliability and soft computing.

Dr. Prashant Johri working as a professor in school of computing science and Engineering, Galgotias University, Gr. Noida. .He received his Ph.D degree in Software Reliability from Jiwaji University Gawalior, India. He has more than 15 years of experience in teaching. He has published numerous papers in the area of software reliability in international journals and conference proceedings His area of research is software reliability, soft computing, parallel distribution and information security.

Dr. Udayan Chanda is currently working as Assistant Professor in Department of Management, Birla Institute of Technology & Science (BITS) Pilani. Earlier he was associated with Industrial Statistics Lab., Department of Information & Industrial Engineering Yonsei University as Post-Doctoral Fellow and Department of Operational Research, University of Delhi as Assistant Professor (Ad-hoc). He received his Ph.D. degree in Marketing Models and Optimization (Operational Research) from University of Delhi, Delhi. He has published numerous papers in the area of Marketing Models, Optimization, Software Reliability and Inventory Management in international journals and conference proceedings. His current research interests include Marketing Models, Inventory Modeling, Software Reliability Growth Modeling, and Dynamic Optimization Techniques.