International Journal of New Computer Architectures and their Applications (IJNCAA) 4(4): 193-200
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2220-9085)

Testing and Debugging Resource Allocation for Fault Detection and Removal Process

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

ABSTRACT

In software development life cycle (SDLC) testing is very important step. One of the key elements of software quality is testing. Fault detection and removal process is also very important when we are doing testing. In the last 30 years numerous software reliability growth models where developed for fault detection and correction process. Majority of models where developed under static condition. The main goal of this article is to examine the resource allocation plan for fault detection and correction process of the software to control the cost during testing and operational phase. For achieving this we developed a model for fault detection and correction process. For solving this model we use Pontryagain's Maximum principle. For optimally resource allocation we use Differential Evolution (DE). Differential Evolution (DE) is an optimization algorithm. A numerical example is also explained for resource allocation for fault detection and removal process.

KEYWORDS

SRGM, Testing Effort Allocation, Correction-Removal Process, Optimal Control Theory, Differential Evolution (DE)

1 INTRODUCTION

For the last few decades it has been perceived that computer has been widely used in difficult kind of problem solving. Lot of complicated and life critical work is done by software systems. So for this kind of software we have to develop quality software. A fault in the software system may produce huge loss of money as well as time. There has been numerous software failures occurred in last three decades resulting in huge financial and other kind of losses. Hence, it is essential for the firm to develop software product that should be error free, reliable and should be suitable for market condition.

Software testing is the process of exercising a system or program with the intent of finding bugs. It is the method of increasing the confidence that the software is free of flaws. But the main problem in testing software is that it cannot be made 100% bugs free. This is not because programmers are irresponsible or careless, but because the nature of software code is complex and humans have only limited ability to manage complexity. Therefore, although testing helps in assessing and improving software quality, but it cannot be performed indefinitely. Hence, testing time is one of the major factors that have to be considered during the testing phase.

Another reason which makes testing phase important is that it consumes major part of the resources available for the software development. Basically, testing activities account for 30 to 50 percent of labor expended to produce a working program [1]. Therefore, it is very important to devote the limited testing-resource efficiently.

For static allocation testing and debugging effort for all the time interval is fixed but in reality it's not true because some error is simple error and some error is complex error. So for simple error it will take less effort and for complex error it will take more effort. For achieving this we developed mathematical model for dynamic allocation of effort allocation we use Pontryagain's Maximum principle for solving the model and Differential Evolution (DE) for optimally allocation of resources. DE is a simple powerful population based stochastic search technique for solving global optimization problems. DE has only a few control variables which remain fixed throughout the optimization process, which makes it easy to implement. A numerical example is also explain for resource allocation.

The paper is subdivided into the following sections. Section two describe the related work of this research, three and four describe the model for fault detection and correction processes, and the cost optimization modeling. In section five, we discuss the solution Procedure. Section six discusses the basic parameter for differential evaluation. A numerical example for fault detection and correction process Vs time using differential evaluation is given in section seven. Finally, in section eight, we conclude our paper with a discussion on results and findings.

2 RELATED WORKS

With the speedy growth in software size and complexity, most software systems today are composed of a number of different modules. During the testing phase, all the testing activities of different stage are challenging for the limited testing-resource. Thus, a difficult problem is how to allocate the total available testing-resource among software modules in an optimal way so that we can finish testing on specified testing limit. [2] Have discussed an overview of the methods that have been developed since 1977 for solving numerous reliability optimization problems including testing-resource allocation problems. [3] Formulated and solved two resource allocation problems for software-module testing, considering the mean number of remaining faults in the software modules. When software testing is finished, the exact number of remaining faults may turn out to be much larger than the mean, and hence [4] studied a dynamic resource allocation strategy for software module testing. This approach takes into account the variations of the number of detected faults throughout testing, re-estimates the model parameters using all the available fault detection data and dynamically allocates the testing resources to the software modules. [5] Discussed optimization problem for testing resource allocations and for the software systems having modular structure and assumed that testing effort allocation is depends upon the size and severity of fault. Numerous SRGM has been proposed in last 4 decades mostly fault detection models based on the assumptions that detection and correction of faults is done concurrently. But, in reality there is always been a time gap ([6], [7], [8], [9] and [10]). [11] Discussed the association between the number of faults deleted with respect to testing effort and/or time. The authors proposed that throughout the testing phase of an SDLC, faults are removed in two stages. First a fault occurred and then the fault causing that failure is corrected; hence the testing effort must be consumed on two distinct processes; failure detection and failure correction. In their article, 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. [12] Solved testing resource allocation problem maximized the number of faults removed from each module under constraint on budget.

[13] Investigated an optimal resource allocation problem in modular software systems during testing phase. The main goal is to minimize the cost of software development when the number of remaining faults is to minimize and a desired reliability objective is given. Authors analyzed the sensitivity of parameters of proposed software reliability growth models. In addition, they also see the impact on the resource allocation problem if some parameters are either overestimated or underestimated. Authors evaluated the optimal resource allocation problems for various conditions by examining the behavior of the parameters with the most significant influence which have been stated by solved numerical example.

[14] Considers two categories of software testing resource allocation problems. The first problem is to minimize the total number of residual faults, given a fixed quantity of testing-effort, and a reliability objective. The last problem is to optimize the amount of software testing effort given the total number of remaining faults in software, and a desired reliability objective. Author has also proposed several strategies for module testing to help software project managers

to solve these types of problems, and make the good decisions. Author provides several systematic solutions based on a nonhomogeneous Poisson process model, allowing efficient allocation of a specified amount of testing resource expenditures for each and every software module under some specified constraints. Author also describes numerical examples on the optimal testing resource allocation problems and performed sensitivity analysis.
[15] Discussed optimal resource allocation plan to minimize the software cost throughout the testing phase and operational phase under dynamic condition using genetic algorithm (GA) technique. Author has been developed mathematical model for allocating testing and debugging resource.
[16] Proposed optimal resource allocation to minimize the software cost during testing and operational phase using optimal control theory. [17] Discussed resource allocation for testing and debugging in dynamic condition. For this authors developed a mathematical model for developing mathematical model author assume when a fault is detected that time the fault is corrected. Differential Evolution (DE) is used for optimally allocation of testing and debugging resource. [18] Discussed Resource control and resource maintenance during the software testing. During the software testing many of the resources like time, effort and budget are consumed. In this paper author proposed an imperfect debugging SRGM during testing and resource allocation is done based on optimizing the effort and reliability. Author used two testing resource allocation schemes one by minimizing the number of remaining faults and allocating the resources to attain the maximum reliability. An experimental result also shows the proposed model well fitted for software testing.

Notations Used:

T	:	The time period.
‘a’	:	Initial number of fault content 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 given point of time‘t’.
$f_i(t)$	:	Recognized 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 $w_3(t)$.
$c_2(f_r(t), w_2(t))$	:	Cost per unit at time ‘t’ for cumulative fault corrected is $f_r(t)$ debugging efforts $w_2(t)$
c_3	:	Budget of testing per unit effort at time ‘t’.

3 MODEL DEVELOPMENT

Fault Detection and Correction Processes Modeling:

For perfect deletion 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. May be it will take time for understanding the type of fault. Hence, after identifying the fault, the debugging people ask the developer to resolve the fault. Thus, there must be a time lag between fault identification and fault 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 correction processes are done in two stages. To begin with, let’s consider ‘a’ is the total number of fault content in the software. Therefore, it is necessary to assume the following equation of fault identification and deletion process;

$$x(t) = \frac{df_d(t)}{dt} = b_1 w_1(t)(a - f_d(t))$$
and
$$y(t) = \frac{df_r(t)}{dt} = b_2 w_2(t)(\mathrm{f}_d(t) - f_r(t))$$
where
$$f_d(0) = 0, f_r(0) = 0 \qquad (1)$$

4 COST OPTIMIZATION MODELLING

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

$$\min \int_0^T \left[c_1(t) x(t) + c_2(t) y(t) + c_3 w_1(t) \right] dt$$
subject to
$$x(t) = \frac{df_i(t)}{dt} = b_1 w_1(t)(a - f_i(t))$$
$$y(t) = \frac{df_r(t)}{dt} = b_2 w_2(t)(\mathrm{f}_i(t) - f_r(t)) \qquad (2)$$
where
$$f_i(0) = 0, \mathrm{f}_r(0) = 0,$$
$$\mathrm{c}_1(t) = c_1\left(f_i(t), w_3(t)\right), \mathrm{c}_2(t) = c_2\left(f_r(t), w_2(t)\right) \text{ and}$$
$$w_1(t) + w_2(t) + w_3(t) = w$$
$$\left(w_1(t); w_2(t); w_3(t)\right) \geq 0$$

5 SOLUTION PROCEDURE

To solve the problem for equation (2), Pontryagain's Maximum principle is applied. The Hamiltonian function is as follows [19]:

$$H(\mathrm{f}_i(t), \mathrm{f}_r(t), \lambda(t), w_1(t), w_2(t), w_3(t), t) = -\left[c_1(t) x(t) + c_2(t) y(t) + c_3 w_1(t)\right] + \lambda(t) x(t) + \mu(t) y(t) \qquad (3)$$

$\lambda(t)$ and $\mu(t)$ are the adjoint variables, which fulfills the following differential equation.

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

And;

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

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 $f_i(t)$ 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 essential conditions hold for an optimal solution:

$$H_{w_1} = 0 \Rightarrow -c_{1w_1}(t) x(t) - (c_1(t) - \lambda(t)) x_{w_1}(t) - c_3 = 0 \qquad (6)$$

$$H_{w_2} = 0 \Rightarrow -c_{1w_2}(t) x(t) - c_{2w_2}(t) y(t) - (c_2(t) - \mu(t)) y_{w_2}(t) = 0 \qquad (7)$$

Where;

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

$$y_{w_2}(t) = \frac{\partial y(t)}{\partial w_2(t)} \tag{8}$$

The additional optimality conditions for Hamiltonian maximization are;

$$H_{w_1 w_1} < 0 \quad and \; H_{w_1 w_1} H_{w_2 w_2} - H_{w_1 w_2} > 0 \tag{9}$$

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

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

And;

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

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

$$w_3^* = w + w_1^* + w_2^* \tag{12}$$

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

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

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

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

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

$$\overset{\bullet}{w}_1 H_{w_1 w_1} + \overset{\bullet}{w}_2 H_{w_1 w_2} + x(t) H_{w_1 f_i} + y(t) H_{w_1 f_r} = -\overset{\bullet}{\lambda} x_{w_1} \tag{15}$$

Time derivative of equation (7) implies;

$$\overset{\bullet}{w}_1 H_{w_2 w_1} + \overset{\bullet}{w}_2 H_{w_2 w_2} + x(t) H_{w_2 f_i} + y(t) H_{w_2 f_r} = -\overset{\bullet}{\mu} y_{w_2} \tag{16}$$

Where;

$$H_{w_1 w_1} = \frac{\partial^2 H(t)}{\partial^2 w_1(t)}$$

$$H_{w_1 w_2} = \frac{\partial^2 H(t)}{\partial w_1(t) \partial w_2(t)}$$

$$H_{w_2 w_2} = \frac{\partial^2 H(t)}{\partial^2 w_2(t)}$$

$$H_{w_i f_r} = \frac{\partial^2 H(t)}{\partial w_i(t) \partial f_r(t)}$$

$$H_{w_i f_i} = \frac{\partial^2 H(t)}{\partial w_i(t) \partial f_i(t)}, \text{ for } i = 1, 2.$$

To solve the above optimization problem, we used Differential Evolution. Differential Evolution is a computerized search and heuristic optimization method for solving difficult type of problem which cannot be solved easily by general methods. Differential Evolution optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its easy formulae, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand.

6 DIFFERENTIAL EVOLUTION

Differential evolution (DE) is a technique that optimizes a problem by iteratively trying to improve a candidate solution with regard to a assumed measure of quality. Such techniques are

commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as Differential evolution (DE) do not guarantee an optimal solution is ever found. Differential evolution is a population-based evolutionary algorithm.
Proposed by [20], is a simple powerful population based stochastic search technique for solving global optimization problems. Differential Evolution has been successfully used in various fields such as communication [21], [22] and mechanical engineering [23] to optimize non convex, non-differentiable and multi modal objective functions. DE has many striking properties compared to other evolutionary algorithms such as, implementation ease, fast convergence rate, the small number of control parameters, and robust performance. DE has only a few control variables which remain fixed throughout the optimization process, which makes it easy to implement. Moreover, DE can be implemented in a parallel processing framework, which enables it to process a large number of training instances efficiently. These properties of DE make it an ideal candidate for the current task of learning a ranking function for information retrieval, where we must optimize non convex objective functions.

The general structure of the Differential Evolution (DE) algorithm looks like that of most other population-based searches. DE keeps parallel version of two arrays, each of which holds population of NP and a D-dimensional, real-valued vectors. The primary array holds the current vector population while the secondary array accumulates vectors that are selected for the next generation.
The main step of the Differential evolution (DE) algorithm is given below:

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

7 NUMERICAL SOLUTION:

Using DE approach, in this section we discuss the numerical solution of the problem discussed in section 3 to find the total number of fault detected and corrected in a given point of time 't' for solving this problem we used MATLAB 7.4.0. [24] The simulation parameter values for solving this problem is as follows:

Parameters	**Values**
Total number of population Size:	200
Total number of generation:	100
Crossover constant (CR):	0.7
Differentiation constant (F):	0.8
a	200
b_1	0.3
b_2	0.3
w_1	0.38
w_2	0.40
$\lambda(0)$	100
$\mu(0)$	100
$f_d(0)$	0
$f_r(0)$	0
c_3	500
c_0	1000
b_0	1000

Table 1 Simulation Parameters

We used above parameters in equation 3 for optimally allocation of fault identified and corrected. The projected number of fault identified and corrected is as follows:

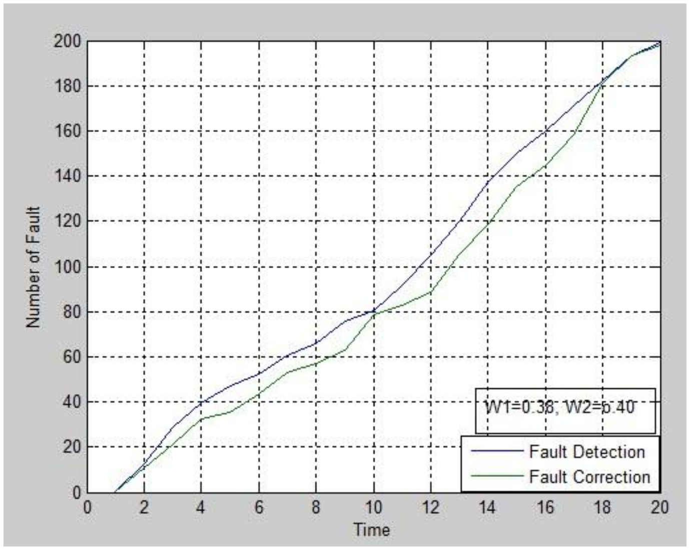

Figure: 1. Number of fault detected and corrected Vs time.

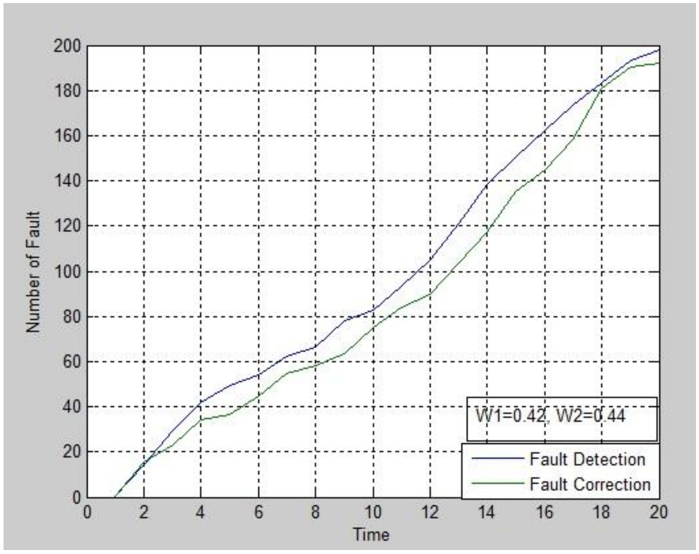

Figure: 2. Number of fault detected and correted Vs time.

Above figure Shows the fault detection and correction process Vs time. In first figure we used W_1= 0.38 and W_2=0.40, in second figure we used W_1=0.42 and W_2=0.44.

8 CONCLUSION

In this paper we developed a mathematical model for detection and removal process of fault. In this study we use Differential Evaluation for dynamic estimation of fault detection and correction process for a given point of time. We have considered fault detection and removal process in two stages and estimated this in MATLAB, we found that the result is better than our static method and also better then Genetic Algorithm. The result described Vs total number of fault detected and corrected. This means the tester and developer can devote their time and resource to finish off their testing and debugging task for well controlled expenditure.

REFERENCES

1. Beizer B., Software Testing Techniques. Boston: International Thomson Computer Press (1990).
2. Kuo W. and Prasad V. R., “An Annotated Overview of System-Reliability Optimization”, IEEE Trans. Reliability, vol. 49(2), pp.176-187 (2000).
3. Ohtera, H., Yamada, S., Optimal allocation and control problems for software-testing resources. IEEE Trans. Reliab. 39 (2), 171–176. (1990).
4. Leung, Y.W., Dynamic resource-allocation for software-module testing. J. Syst. Software 37 (2), 129–139. (1997).
5. Kapur, P.K., Bardhan A.K.and Yadavalli V.S.S., 'On allocation of resources during testing phase of a modular software', International Journal of Systems Science, 38 (6), 493 - 499. (2007)
6. Ohba, M., 'Software reliability analysis models', IBM Journal of research and Development 28, 428-443 (1984).
7. Schneidewind, N.F., 'Analysis of error processes in computer software', Sigplan Notices 10, 337-346 (1975).
8. Xie, M. and Zhao, M., ' The Schneidewind software reliability model revisited.' Proceedings of the 3rd International Symposium on Software Reliability Engineering, 5, 184-192 (1992).
9. Schneidewind, N.F., 'Analysis of error processes in computer software', Sigplan Notices 10, 337-346 (1975).
10. Gokhale, S.S., Wong, W.E., Trivedi, K.S. and Horgan, J.R., 'An analytic approach to architecture-based software reliability prediction' in Proceedings of the International Symposium on Performance and Dependability, September, pp. 13-22 (1998).
11. Kapur, P.K., and Bardhan, A.K.., ‘Testing Effort Control Through Software Reliability Growth Modelling’, International Journal of Modelling and Simulation, 22, 90–96 (2002).
12. Khan M, Ahmad N, and Rafi L. "Optimal Testing Resource Allocation for Modular Software Based on a Software Reliability Growth Model: A Dynamic Programming Approach, " Proceedings of the International Conference on Computer Science and Software Engineering (2008),

13. Huang, C. Y., Lo, J. H., Kuo, S. Y. and M. R, ―Optimal Allocation of Testing-Resource Considering Cost, Reliability, and Testing-Effort, Dependable Computing, 2004. Proceedings. 10th IEEE Pacific Rim International Symposium, pp. 103-112 (2004).
14. Chin-Yu Huang, and Michael R. Lyu, Optimal Testing Resource Allocation, and Sensitivity Analysis in Software Development, ―IEEE Transactions on Reliability, VOL. 54, NO. 4, (2005).
15. Nasar. Md., Johri Prashant, Udayan Chanda,"Dynamic Effort Allocation Problem Using Genetic Algorithm Approach", IJMECS, vol.6, no.6, pp.46-52, (2014) .DOI: 10.5815/ijmecs.2014.06.06
16. Kapur P.K., Pham Hoang, Chanda Udayan and Kumar Vijay: Optimal allocation of testing effort during testing and debugging phases: a control theoretic approach, International Journal of Systems Science, (2012) DOI:10.1080/00207721.2012.669861
17. Nasar Md, johri prashant and chanda Udayan "A Differential Evolution Approach for Software Testing Effort Allocation" Journal of Industrial and Intelligent Information (JIII), SAN JOSE, CA, USA Volume 1, Issue 2, pp. 111-115, ISSN 2301-3745 (2013)
18. SK.Md.Rafi, and ShahedaAkthar, " Resource Allocation to Software Modules in Software Testing with Imperfect-debugging SRGM, " International Journal of Computer Applications (0975 – 8887) Volume 18– No.2, (2011)
19. Sethi, S.P., and Thompson, G.L., Optimal Control Theory - Applications to Management Science and Economics (2nd ed.), New York: Springer (2005).
20. Price,, K. and Storn R., "Differential evolution a simple evolution strategy for fast optimization," Dr. Dobb's Journal, vol.22, pp. 18 – 24, (1997)
21. Storn, R., "Differential evolution design for an IIR-filter with requirements for magnitude and group delay," Technical Report TR-95-026, International Computer Science Institute, Berkeley, (1995).
22. Angira, R. and Babu, B., "Evolutionary computation for global optimization of non-linear chemical engineering processes," in Proceedings of International Symposium on Process Systems Engineering and Control, Mumbai, , pp. 87–91 (2003).
23. Babu, B. and Angira, R., "Optimization of non-linear functions using evolutionary computation," in Proceedings of the 12th ISME International Conference on Mechanical Engineering, India, pp. 153–157 (2001).
24. Price, K. and Storn, R., (July 2003) Web site of DE. Available: http://www.ICSI.Berkeley.edu/~storn/code.html