International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - ICSCAAIT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

# Genetic Algorithms: Concepts, Applications in Software Engineering and Test Data Generation using GA and Hamming Distance Approach

Manish Saraswat[1], R. C. Tripathi[2]
[1]Associate Professor, Department of MCA, Geetanjali Institute of Technical Studies, Udaipur
[2]Professor, Department of Computer Science & Engineering, Lingaya University, Faridabad
[1]manishsaraswat24@gmail.com, [2]ramesh_c_tripathi@yahoo.co.in

***Abstract:*** ***The basic objective of software engineering is to develop methods and procedures for software development that can scale up for large systems and that can be used consistently to produce high-quality software at low cost and with a small cycle of time. The effective and timely development of software requires the activities of software engineering be automated to large extent and human resource intervention be minimized to optimum level. In this paper, we explore some basic learning concepts of genetic algorithms and take a look at how genetic algorithm (GA) can be used to build tool for software development. We also propose GA and hamming distance based method for test cases identification***

***Keywords: SDLC, Genetic Algorithm, Software Testing, Hamming Distance.***

## I. INTRODUCTION

Software Engineering is systematic approach to development, operation, maintenance and retirement of any software. Development of any software follows a series of steps known as the 'Software Development Cycle' (SDLC). Software testing is an important part of SDLC. Software testing is the major quality control measure employed during the software development. Its basic objective is to detect errors in the software. Testing consumes the highest (approximately 50-60%) time and budget in Software Development Life Cycle. Software Testing is often used in association with the terms 'verification' and 'validation' of complete SDLC [1], [2], [3].

Modern software is becoming more expensive to build and maintain. Software development management and software quality goals are necessary, but not competent for the needs of today's marketplace. Shorter cycle time, completed with least resources is also in demand [5]. The challenge of developing software system in a fast moving Evolutionary Algorithms scenario gives rise to a number of demanding situation.

First situation is identifying software components is a crucial task in software development. The second one is to minimize number of test cases develop for the testing purpose. To answer the challenge, a number of approach can be utilized one such approach is the evolutionary algorithm [4]. By using evolutionary algorithm software is developed, modified and maintained at specification level, and automatically produced high quality software in shorter period [6]. This evolutionary approach will enable software engineering to become the discipline capturing and automating currently undocumented domain and design knowledge [7].

In order to realize its full potential, there are tools and methodologies needed for the various tasks inherent to the evolutionary algorithm. In this paper, we survey the existing work on application of GA in software engineering and provide research directions for the future work in this area.

## II. GENETIC ALGORITHM

Genetic algorithms are search algorithms which are conceptually based on the methods, which the living organism adopts to survive in their living environment. These methods of adaptation are known as natural selection or evolution. The genetic algorithms are heuristic in nature, their performance and output efficiency can vary across the multiple runs. GAs are useful and work efficiently when the search space is large, complex and poorly understood, when domain knowledge is scarce or expert knowledge is difficult to encode. It is also useful when there is a need to narrow the search space and incase of failure of traditional search methods. A simple genetic algorithm is given below:

```
Genetic Al {
initialize population;
evaluate population;
while TerminationCriteriaNotSatisfied
{
select parents for reproduction;
perform
recombination and mutation;
evaluate population;
}
}
```

Genetic Algorithms begin with a set of initial individuals as the first generation, which are sampled at random from the problem domain. Each individual in each generation is evaluated with a fitness function. Based on the evaluation, the evolution of the individuals may approach the optimal

solution. The most common operations of genetic algorithms are designed to produce efficient solution for the target problem. These primary operations include:

*a). Reproduction:* This operation assigns the reproduction probability to each individual based on the output of the fitness function. The individual with a higher ranking is given a greater probability for reproduction. As a result, the fitter individuals are allowed a better survival chance from one generation to the next.

*b). Crossover:* This operation is used to produce the descendants that make up the next generation. This operation involves the following crossbreeding procedures:

i) Randomly select two individuals as a couple from the parent generation.

ii) Randomly select a position of the genes, corresponding to this couple, as the crossover point. Thus, each gene is divided into two parts.

iii) Exchange the first parts of both the genes corresponding to the couple.

iv) Add the two resulted individuals to the next generation.

*c). Mutation:* This operation picks up a gene at random and changes its state according to the mutation probability. The purpose of the mutation operation is to maintain the diversity in a generation to prevent premature convergence to a local optimal solution. The mutation probability is given intuitively since there is no definite way to determine the mutation probability.

Upon completion of crossover processing and mutation operations, there will be an original parent population and a new offspring population. A fitness function should be devised to determine which of these parents and offsprings can be survived into the next generation. After performing the fitness function, these parents and offsprings are filtered and a new generation is formed. These operations are iterated until the expected goal is achieved.

## III. GENETIC ALGORITHMS APPLIED IN SOFTWARE ENGINEERING

Several areas in software development have already witnessed the use of GAs. In this section, we take look at some reported result of application of GAs in the field of software engineering. A variety of life cycle models has been proposed and is based on task involved in developing software [8].Figure 1 shows SDLC/CBSD phases and applications of GAs in software engineering.

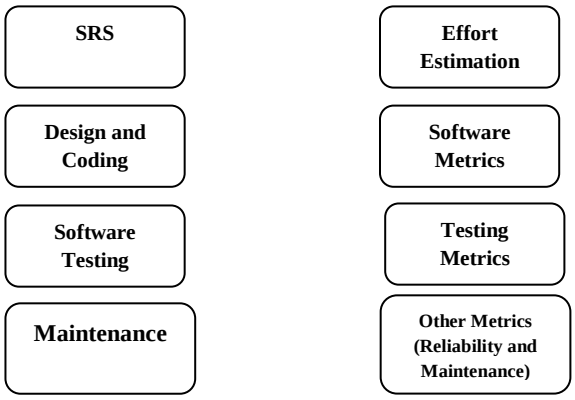


Fig. 1. SDLC/CBSD Phases and Applications of GAs

*1. Software Project Effort Estimation:*

Software cost estimation is one of the most challenging issues in software project development. To produce the accurate estimation, many models have been developed, but no model proves efficient with the uncertainty of the project development. Most of these models are based on the size measure, such as Lines of Code (LOC) and Function Point (FP) and Size estimation accuracy directly effect on cost estimation accuracy.

As all we know the COCOMO model is the important model for Software Cost Estimation. Today's effort Estimation models are based on soft computing techniques such as, genetic algorithm, fuzzy logic, neural network etc for finding the accurate predictive software development effort and time estimation. Genetic Algorithm can provide significant enhancement in accuracy and has the potential to be a valid additional tool for software effort estimation in large project. Genetic algorithm has been used for difficult numerical optimization problems and also used to solve system identification, signal processing and path searching problems [29].

Brajesh et al. proposed a model to estimate the software effort for projects sponsored by NASA using binary genetic algorithm. Modified version of the COCOMO model was provided to consider the effect of methodology in effort estimation. The performance of the developed model was tested on NASA software project data and the developed models were able to provide good estimation capabilities [30].

Vishaliet et al. proposed algorithm (GAs) was tested and the obtained results were compared with the ones obtained using the current COCOMO model coefficients. The results of the experiment show that in most cases the results obtained using the coefficients optimized by the proposed algorithm are close to the ones obtained using

the current coefficients. Comparing organic and semi-detached COCOMO model modes, it can be stated that use of the coefficients optimized by the GA and ACO in the organic mode produces better results in comparison with the results obtained using the current COCOMO model coefficients [28].

*2. Software Metrics (Design and Coding):*

Software metrics are numeric value related to software development. Metrics have traditionally been consisting through the definition of an equation, but this technique is limited by the fact that all the interrelationships among all the parameters be fully understood. The aim of research is to find the alternative methods for generating software metrics. Deriving a metrics using a GAs has several advantages [15].

R Vankudoth et et al. work on selection of system software component. It is an important decision of design stage and has a significant impact on various system quality attributes. To determine system software component based on architectural style selection, the software functionalities have to be distributed among them components of software the Genetic Algorithm can be used to identify software components and their responsibilities. [34].

K Vijayalakshmi et et. al has given an automated approach based on Genetic Algorithm that enables the selection of software components both considering functional and non-functional requirements to find the best combination of components [9], [10], [11], [12].

Seyed Mohammed et al. proposed a novel GA-based algorithm as a powerful optimization search algorithm, called SCI-GA (Software Component Identification using Genetic Algorithm), to identify components from analysis models.

Kwonget.et al. has given the formulation of an optimization model of software components selection for CBSS development. This model has two objectives: maximizing the functional performance of the CBSS and maximizing the cohesion and minimizing the coupling of software modules. A genetic algorithm (GA) is used to solve the optimization model for determining the optimal selection of software components for CBSS development. Saxsena et al. an attempt to throw light which on the one of the major issue of component based software engineering is concerned with the "Component Selection". Genetic Algorithms based approach is used for component selection to minimize the gap between components are selected [17].

*3. Software Testing Activities:*

Software testing is an important part of SDLC. Testing is the major quality control measure employed during the software development. Its basic objective is to detect errors in the software. Testing consumes the highest (approximately 50-60%) time and budget in Software Development Life Cycle. Software Testing is often used in association with the terms 'verification' and 'validation' of complete SDLC [1], [2], [3].

Test cases and test data generation is the key problem in software testing and as well as its automation improves the efficiency and effectiveness and lowers the high cost of software testing. Generation of test data using random, symbolic and dynamic approach is not enough to generate optimal amount of test data.. That why there is need for generating test data using search based technique. In addition to these there is also need of generating test cases that concentrate on error prone areas of code [16], [17], [18], [19].

The process of test data generation involves activities for producing a set of test cases that satisfy a chosen testing criterion. Horgan et. al. has previously shown that test cases selected on the basis of test adequacy criteria are more effective at discovering the defects in software under the test. Although it is possible to manually generate effective the test cases but the more cost effective approach is to automate the test data generation with ensuring that the given criteria is met Mansour Nasrat and Salame Miran, The test objectives are expressed numerically and used subsequently to formulate a suitable fitness function that evaluates the suitability of generated test cases. Many attempts [Michael Christoph et. al.],[Mc Minn Phil],[Birt J R and Sitte R ] [Patton et. al. ] [Li Huaizhong and Lam C Peng, ],[ Rajappa et. al.] have been made over the years to develop such a tool to generate test data automatically. Lin and Yeh show that level of coverage is increased using Hamming distance from the branch testing to path. They extend the definition of hamming distance from first order to $n^{th}$ order (n>1) for measuring the distance between two paths (Extended Hamming Distance (EHD).

Wegener et. al. developed fully automatic GA- based test data generation for structural testing , specially statement and branch coverage of real word embedded software systems. The proposed fitness function Wegener et. al consist of two major building blocks: approximation level and normalized predicate local distance. Overall fitness value is summation of both values. berndt et. al. describes the preliminary results from a GA based approach to software test case breeding. Andre Baresel et. al. work to enhance the constriction of fitness function in order to improve the evolutionary testability by obtaining the higher coverage with less number of resources. Rajappa

et. al. proposed a solution with combination of graph theory and genetic algorithm.

Yang et. Al introduces an approach of generating test data for a specific single path based on genetic algorithms. The similarity between the target path and execution path with sub path overlapped is taken as the fitness value to evaluate the individuals of a population and drive GA to search the appropriate solutions. The experimental results prove that the function performs better as compared with the other two typical fitness functions for the specific paths employed by the authors [22], [23].

Aladeen et al.[17] have compared the software test data for automatic path coverage using genetic algorithm with Yong [20] for generating test data of path testing. They found GAs is useful in reducing the time required for lengthy testing by generating the meaningful test cases for path testing. The GAs is required to be built for structural testing for reduce execution time by generating more suitable test cases.

*4. Other Software Metrics (Quality, Reliability and Maintenance):*

Garvin describes quality from five different views: transcendental view, user view, manufacturers view, product view and value based view. Quality must be monitored from the early phases to final phase such as analysis, design, implementation and maintenance phases.

M Amoui et al. work for Improving software quality. It is a major area in software development process. Despite all previous attempts to evolve software for quality improvement, these methods are neither scalable nor fully automatable so in this research authors approach software evolution problem by reformulating it as a search problem. For this purpose, author apply software transformations in a form of GOF patterns to UML design model and evaluated the quality of the transformed design according to Object-Oriented metrics particularly 'Distance from the Main Sequence'. The implementation results show that Genetic Algorithm is able to find the optimal solution efficiently, especially when different genetic operators, adapted to characteristics of transformations, are used [40].

D M Thakore et al. work on the security issues by using GAs. Assigning access specifier is not a n easy task as it decides over all security of any software though there are many metrics tools available to measure the security at early stage. Objective of Secure Coupling Measurement Tool (SCMT) is to generate all possible solutions by applying Genetic Algorithm (GA). It is quietly different than any other security Measurement Tool because it filters input design before applying metrics by GA.SCMT uses coupling, also feature of OO design to determine the security at design level.. [35].

S H Aljahdali use GAs as powerful technique to estimate the parameters of well known reliability model. Software reliability models are useful to estimate the probability of the software fail along the time. Several different models have been proposed to predict the software reliability growth (SRGM); but none of them has proven to perform well considering different project characteristics. The ability to predict the number of faults in the software during development and testing phases. GAs is a powerful machine learning technique and optimization techniques to estimate the parameters of well-known reliably growth models. Moreover, machine learning algorithms, proposed the solution to overcome the uncertainties in the modeling by combining multiple models aiming at a more accurate prediction at the expense of increased uncertainty [36].

Maintenance is an important activity in the software development life cycle and no software product can do without undergoing the process of maintenance.

Abdulrahman et.al. proposes an Evolutionary Neural Network (NN) model to predict software maintainability. The proposed model is based on a hybrid intelligent technique wherein a neural network is trained for prediction and a genetic algorithm (GA) implementation is used for evolving the neural network topology until an optimal topology is reached and the model was applied on a popular open source program, namely, Android. The results are very fine, where the correlation between actual and predicted points reaches 0.91 [37].

## IV. TEST DATA GENERATION USING GA AND HAMMING DISTANCE APPROACH

This proposed study gives some details of basic steps for test data generation using genetic algorithm and hamming distance method. The proposed approach has been tested on specific code which determines the power of input number when base and exponent are specified. The pseudo code of power generation program is represented below and here the target value is initially bounded:

```
1. double powerxy( int x, int y)
{
2. pow=1;
3. for( int j=1 ;  j!=(y+1)  ; j++)
4. {
5.    Pow= pow *x;
6. }
7. Return pow;
8  }
```

Fig. 2. Pseudo Code for Power Generation Program

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - ICSCAAIT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

The value is converted into binary form and appropriate weights are calculated as:

Target Value 1016

Target in Binary: 00001111111000

Weight -8.192 -4.096 -2.048 -1.024 0.512 0.256 0.128 0.064 0.032 0.016 0.008 -0.004 -0.002 -0.001

Now fitness value of each individual is calculated. The value of individual is compared with the target value. The hamming distance is also calculated, which then multiplied by respective weight at the bit position. The summation of all these differences (weight) is then calculated as shown in Figure 3 and Figure 4 :

New individual x=5, y=3
Output Value=5.0
Output in Binary: 00000000000101
Target Value: 00001111111000
Hamming distance: 00001111110011
Difference Value: 1.011
Output Value=25.0
Output in Binary: 00000000011001
Target Value: 00001111111000
Hamming distance: 00001111011111
Difference Value: .991
Output Value=125.0
Output in Binary: 00000001111101
Target Value: 00001111111000
Hamming distance: 00001111011111
Difference Value: .891

Fig. 3. Calculation of Fitness Function

New individual x=6 y=5
Output Value=6.0
Output in Binary: 00000000000110
Target Value: 00001111111000
Hamming distance:00001111110010
Difference Value: 1.010
Output Value=36.0
Output in Binary: 00000000100100
Target Value: 00001111111000
Hamming distance: 00001111010100
Difference Value: .980
Output Value=1296.0
Output in Binary: 00010100010000
Target Value: 00001111111000
Hamming distance: 00000100011000
Difference Value: -0.280

Fig. 4. Calculation of Fitness Function for New Individual

Next step is to select the individuals for next generation. The actual selection is done on the basis of roulette wheel concept as calculated and shown in table 1:

Table 1. Selection of Individuals in Next Generation

| x | y | Expected Count | Count |
|---|---|---|---|
| 3 | 5 | 1.5314 | 2 |
| 2 | 6 | 0.4030 | 0 |
| 2 | 2 | 0.0247 | 0 |
| 4 | 4 | 1.6133 | 2 |
| 5 | 3 | 0.7878 | 1 |
| 6 | 3 | 1.3611 | 1 |
| 2 | 8 | 1.6133 | 2 |
| 3 | 5 | 1.5314 | 2 |
| 9 | 2 | 0.5096 | 0 |
| 8 | 3 | 3.2279 | 3 |
| 7 | 2 | 0.3081 | 0 |
| 1 | 8 | 0.0052 | 0 |
| 12 | 1 | 0.0754 | 0 |

Now the cross over is applied on any two which are selected at random from Table 2 and the individuals, which do not take part in crossover are also considered in next generation. The individuals which are selected for

crossover is shown in Table 2 and the individuals which are not selected for cross over are shown in Table 3:

Table 2. Crossover on Selected Individuals

| x | y | Before Crossover | | After Crossover | | Site |
|---|---|---|---|---|---|---|
| 8 | 3 | 0001000 | 0000011 | 0001000 | 0000100 | 11 |
| 4 | 4 | 0000100 | 0000100 | 0001000 | 0000011 | 11 |
| 8 | 3 | 0001000 | 0000011 | 0001000 | 0001000 | 7 |
| 2 | 8 | 0000010 | 0001000 | 0000010 | 0000011 | 7 |
| 3 | 5 | 0000011 | 0000101 | 0000010 | 0001000 | 9 |
| 2 | 8 | 0001000 | 0001000 | 0000011 | 0000101 | 9 |
| 8 | 3 | 0001000 | 0000011 | 0000100 | 0000100 | 12 |
| 4 | 4 | 0000100 | 0000100 | 0001000 | 0000011 | 12 |
| 3 | 5 | 0000011 | 0000101 | 0000011 | 0000101 | 5 |
| 3 | 5 | 0000011 | 0000101 | 0000011 | 0000101 | 5 |

Table 3. Individuals not Participated in Crossover

| x | y | Individuals |
|---|---|---|
| 3 | 5 | 00000110000101 |
| 5 | 3 | 00001010000011 |
| 6 | 3 | 00001100000011 |

Now mutation is applied over the individuals (Table 4) by exchanging the values randomly.

Table 4. Individuals Participated in Mutation

| x | y | Before Mutation | After Mutation | Site |
|---|---|---|---|---|
| 3 | 5 | 00000100001000 | 00000101001000 | 7 |
| 2 | 8 | 00000100000011 | 00000100000111 | 3 |

The above procedure is for one generation and it continues till a success criterion is achieved. Further the generated test cases may be feasible and unfeasible test cases. Feasible test cases traverse the program with desired path and provide desired output, this example some feasible test cases are listed in Table 5:

Table 5. List of Some Feasible Test Cases

| x | y | Power(Value) |
|---|---|---|
| 6 | 5 | 7776 |
| 8 | 4 | 4096 |
| 2 | 10 | 1024 |
| 4 | 5 | 1024 |

**Table 5 List of some feasible Test Cases**

While unfeasible test cases are traverse the program in undesired way, for example some unfeasible test cases may be:

Table 6. Some Unfeasible Test Cases

| x | Y |
|---|---|
| 5 | 0 |
| 8 | 0 |

Table 7. Some Unsatisfied Test Cases

The unsatisfied test cases are those whose actual count comes to zero means they are not going to take part in a crossover and mutation, some of them are shown as Table 7:

| x | Y |
|---|---|
| 2 | 6 |
| 2 | 2 |
| 1 | 8 |
| 5 | 3 |
| 9 | 2 |

## V. CONCLUSION AND RESEARCH DIRECTIONS

In this paper, we show how GAs has been used in tackling many software engineering problems. The GAs has been used in various phases of software development like from requirement and analysis phase and software testing phase. It is also used developing new metrics. This will definitely help maturing software engineering discipline. further test data generation using genetic algorithm and hamming distance method is proposed. The proposed approach has been tested on specific problem and feasible, unfeasible test case are identified. Further test cases classified as satisfied and un satisfied for the problem.

## VI. REFERENCES

[1] Pressman R.S [2005], 'Software Engineering: A Practitioner's Approach', 6th Edition, Tata McGrath.

[2] Prasad [2005] 'Software Testing Tools', Edition, Wiley-dreamtech, India.

[3] Goldberg D.E [1989], 'Genetic Algorithms in Search, Optimization and Machine Learning', Addition Wesley Publication New York, NY.

[4] H. Seyed, M Hossein, and S Jalili, "SCI-GA: Software Component Identification using Genetic Algorithm", Journal of Object Technology, 2013.

[5] K Vijayalakshmi, N Ramaraj, and RAmuthakkannan, "Improvement of component selection process using genetic algorithm for component-based software development", International Journal of Information Systems and Change Management, 2008, pp. 63-80.

[6] Y Singh, P K Bhatia, A Kaur, and O Sangwan, "Application of neural networks in software engineering: A review", In International Conference on Information Systems, Technology and Management.

[7] M Harman, S AMansouri, and Y Zhang, "Search based software engineering: A comprehensive analysis and review of trends techniques and applications", Department of Computer Science, King's College London, Tech. Rep. TR-09-03, 2009.

[8] M R Girgis, "Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm", J. UCS 11, 2005, PP.898-915.

[9] G M Morris, D S Goodsell, R S. Halliday, Ruth Huey, William E Hart, R K Belew, and A J Olson, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function", Journal of computational chemistry 1998.

[10] H Mühlenbein, and D S Voosen, "Predictive models for the breeder genetic algorithm i. continuous parameter optimization", Evolutionary computation.

[11] J F Tang, L F Mu, C K Kwong, and X G. Luo, "An optimization model for software componentselection under multiple applications development", European Journal of Operational RES.

[12] J Pande, C J Garcia, and D Pan, "Optimal component selection for component based software development using pliability metric", ACM SIGSOFT Software Engineering Notes 38, no. 1, 2013, pp. 1-6.

[13] A Dixit, and P. C. Saxena, "Software component retrieval using genetic algorithms", In Computer and Automation Engineering, 2009. ICCAE'09. International Conference on, IEEE, 2009, pp. 151-155.

[14] S Parnami, K S Sharma, and S V Chande., "A survey on generation of test cases and test data usingartificial intelligence techniques", International Journal of Advances in Computer Networks and its S

[15] A S Ghiduk, and M R Girgis, "Using genetic algorithms and dominance concepts for generating reduced test data", Informatica 34, no. 3 2010.

[16] D C Koboldt, K M Steinberg, D E Larson, R K. Wilson, and E R. Mardis, "The next-generation sequencing revolution and its impact on genomics", Cell 155, no. 1, 2013, pp.27-38.

[17] S M Mohi-Aldeen, R Mohamad, and S Deris, "Automatic Test Case Generation for Structural Testing Using Negative Selection Algorithm", 2009.

[18] V Rajappa, ABiradar, and S Panda. "Efficient software test case generation using genetic algorithm based graph theory", In 2008 First International Conference on Emerging Trends in Engineering and Technology, IEEE, 2008, pp. 298-303.

[19] Y Fuqing, M Hong, and LKeqin, "Software Reuse and Software Component Technology [J]", Acta Electronica Sinica 2, 1999.

[20] S Sabharwal, R Sibal, and C Sharma, "Prioritization of test case scenarios derived from activity diagram using genetic algorithm", In Computer and Communication Technology (ICCCT), 2010 International Conference on, IEEE,2010, pp. 481-485.

[21] R P Pargas, M J Harrold, and R R Peck, "Test-data generation using genetic algorithms", Software Testing Verification and Reliability 9, no. 4 1999

[22] C Sharm, S Sabharwal, and R Sibal, "A survey on software testing techniques using genetic algorithm".

[23] A Kaur, and S Goyal, "A genetic algorithm for regression test case prioritization using code coverage", International journal on computer science and engineering 3, no. 5, 2011, pp. 1839-1847.

[24] R Krishnamoorthi, and SA S A Mary, "Regression test suite prioritization using genetic algorithms", International Journal of Hybrid Information Technology 2, no. 3, 2009, pp.35-52.

[25] Reena, Pradeep Kumar Bhatia Application of Genetic Algorithm in Software Engineering: A Review International Refereed Journal of Engineering and Science (IRJES) ISSN (Online) 2319-183X, (Print) 2319-1821 Volume 6, Issue 2 (February 2017), PP. 63

[26] P McMinn, "Search-based software test data generation: A survey," Software Testing Verification and Reliability 14, no. 2, 2004, pp.105-156.

[27] K Singh, "Effective Software Testing using Genetic Algorithms", Journal of Global Research in Computer Science 2, no. 4, 2011.

[28] N Haghpanah, S Moaven, J Habibi, M Kargar, and S H Yeganeh, "Approximation algorithms for software

component selection problem", In 14th Asia-Pacific Software Engineering Conference

[29] S Bhatia, A Bawa, and V K Attri, "A Review on Genetic algorithm to deal with Optimization of Parameters of Constructive Cost Model", International Journal of Advanced Research in Computer and Communication Engineering 4, no. 4, 2015.

[30] B K Singh and A. K. Misra, "Software effort estimation by genetic algorithm tuned parameters ofmodified constructive cost model for nasa software projects", International Journal of Computer App.

[31] A Dhiman and C Diwaker ,"Optimization of COCOMO II effort estimation using genetic algorithm", American International Journal of Research in Science, Technology, Engineering & Mathematics 3, no.2, 2013.

[32] I Maleki, A Ghaffari and M Masdari, "A new approach for software cost estimation with hybrid genetic algorithm and ant colony optimization", International Journal of Innovation and Applied.

[33] R Vankudoth, P Shireesha and T. Rajani, “A Model of System Software Components Using Genetic Algorithm and Techniques”, International Journal of Advanced Research in Computer Science and Software Engineering, 2016, pp. 301-306.

[34] A Martens, H Koziolek, S Becker, and R Reussner, "Automatically improve software architecture models for performance, reliability and cost using evolutionary algorithms, ACM, 2010, .

[35] J. A McCall, P. K, Richards and G. F. Wallers, “Factors in software quality “, Griffiths Air Force Base N. Y: Rome Air Development Center Air Force Systems Command, 1977.

[36] M Amoui, S Mirarab, S Ansari, and C Lucas, "A genetic algorithm approach to design evolution using design pattern transformation", International Journal of Information Technology and Intelligent Computing 1, no. 2, 2006, pp. 235-244.