A NOVEL APPROACH FOR PRIORTIZATION OF OPTIMIZED TEST CASES

Abhishek Singhal
Amity School of Engineering and Technology
Amity University
Noida, India
asinghal1@amity.edu

Swati Chandna
Amity School of Engineering and Technology
Amity University
Noida, India
swatichandna111@gmail.com

Abhay Bansal
Amity School of Engineering and Technology
Amity University
Noida, India
abansal1@amity.edu

Abstract—**Generation and prioritization of test cases is one of the major issue in software testing.Maximum number of faults are identified through test cases only. Clients confidence can be gained through software testing. This paper firstly generates test cases using decision coverage metrics which gave redundant set of set cases. So, in in order to generate optimized set of test cases, genetic algorithm is used so that we can generate maximum number of faults with the minimum number of test cases. After generating the optimized set of test cases we proposed a technique to prioritize those test cases to indicate the order with which test case may be addressed which is known as prioritization of test cases using proposed algorithm which gave results in less time by consuming less resources.**

Keywords- Software Testing, Test Case Generation, Fitness Function, Genetic Algorithm, Optimization, Prioritization

I. INTRODUCTION

Software testing is one of the major components in software development life cycle.Validation and verification process occurs in software testing. Customer confidence can be gained through software testing only. Nowadays, artificial intelligence influences test automation process in the area of software testing. Software testing is very time consuming work where 50% of the software system development resources are spent[3].Test case automation and is one the most powerful aspect in automatic testing.This paper proposes an algorithm which uses genetic algorithm as an optimization technique to generate optimized set of test cases and further a technique is proposed to prioritize the test cases to generate and schedule the order of the test cases, thus giving strong level of software coverage in less time.

II. GENETIC ALGORITHM

Genetic Algorithm is a search based technique to find approximate solution to optimization problem.Mechanics of natural evolution is based on Genetic Algorithm, through their evolution successive generations search for adaptations through which difficult problems can be solved easily giving beneficial results.GA represents a randomized exchange of structured information.It represents Darwin's principal for the survival of the fittest. Each generation consists of individuals called chromosomes. The initial generation is created randomly. Fitness level is acquired by each individual who is basically based on cost of the problem taken into consideration [1].

Genetic operations of crossover, mutation are used to create new offspring population from the initial population. Selection process is done in reproduction in this, selecting individuals from the initial population of chromosomes is done. In crossover two randomly located chromosomes are mated. Then mutation is applied further to the genes which were selected. The process is repeated many times till fitness function is maximized.

III. GENERATING OPTIMIZED SET OF TEST CASES USING GENETIC ALGORITHM

Genetic Algorithm is basically based on the survival of the fittest given by Darwin. It is used to find the optimum solutions. It has very exhaustive search procedure. When GA is used to solve optimization problems best results are obtained. GA is used to search the domain of input values and to find those which satisfy the desired goal of testing. Inputs are combined to generate new inputs which are used for further searching of the desired goal. The evolution is based on two primary operators' crossover and mutation[1]. Despite the randomized nature of Genetic Algorithm, GA is not a simple random search. It uses old knowledge held in parent population to generate new solutions with improved performance. Best solutions are retained and relatively bad ones are discarded.

Initial Population comprises set of individuals generated randomly. The selection of the initial generation is one of the main task which effects the performance of the next generation significantly. Each individual is represented as a chromosome(binary).The chromosome are composed of genes and are modified by applying crossover and mutation operators. Fitness of each individual is calculated to compare the performance of each individual[4]. An individual which is near the optimum solution has a higher fitness value[8]. The process of mutation and crossover are applied to produce new members.

Crossover and Mutation are the operators which are most commonly used in practice. Crossover operates at individual level represented in binary form. This process selects bits from parents and generates new offspring's. In crossover two individuals are selected and a point along the bit string is selected and swapped. Mutation changes random bits in the binary string[6]. In this simply the state of the gene is changed from 0 to 1 or vice-versa[8]. After the crossover and mutation operations, original population and newly generated population is what we get. The survival of parents and offspring depends on the fitness value of every member of the population.

Fitness function is used to calculate the best solution of a problem. This is done by using a fitness function. An individual, who is near optimum solution, gets a higher fitness value than an individual[2]. Search space is represented by every valid fitness value.

Selection operator selects two individuals from a generation to become parents for the recombination process. This selection may be based on fitness value.

IV. ALGORITHM FOR GENERATING TEST DATA

1.GA begins with initial population which is randomly generated where each chromosome is represented as chromosome.
2.Fitness of each individual is calculated.
3.An individual which has a higher fitness value is the near optimum solution.
4.A stopping criteria is decided for stopping test data generation
5.Data is prepared for the next generation[1]

The process will be repeated until the population has evolved to form an optimum solution of the problems or until a maximum number of iterations have taken place. If the fitness function is effective, desired inputs will be selected early and helps to traverse the path[1].

V. TEST CASE PRIORITIZATION

The order with which the test case may be addresses is known as prioritization of test cases. A test case with the highest rank has the highest priority as test case with second highest rank has second highest priority and as so on. There are two ways of test case prioritization i.e. general test case prioritization and version specific test case prioritization. In general test case prioritization, test cases are prioritized that will be useful over a subsequent modified versions of the original program without any knowledge of the modification. In the version specific test case prioritization, test cases are prioritized when the original program is changed to the modified program,

the changes that have been made originally[12]. The simplest priority category scheme is to assign a priority code to every test case, test case with priority code 1 is more important than test case of priority code 2 [12][13]. In this paper, a technique have been developed to prioritize the test cases which prioritizes test cases and recommends use of high priority test cases first and then low priority in descending order.

VI. PRIORITIZATION OF TEST CASES : PROPOSED ALGORITHM

Firstly we, implemented genetic algorithm to optimize the number of test cases. Now we proposed an algorithm to prioritize the number of test cases resulted after minimization of test cases. This technique recommends use of test cases which hold high priority first and then low priority test cases in descending order till the reasonable amount of confidence is achieved. In order to design the technique we need to have Program whose test cases have been optimized using genetic algorithm, Test suite with test cases t1, t2, t3, t4…..tm, number of lines of source code are stored in two dimensional array. To further analyze this technique proposed algorithm is stated below where T is two dimensional array used to store line numbers, scode is used to store total number of lines of source code, nf is one dimensional array used to store the number of lines of source code matched with the lines of source code covered by each test case, pos is a one dimensional array and is used to set the position of each test case when nf is sorted, pt sets the bit to 1 corresponding to the position of the test case to be removed, priority is also a one dimensional array used to store the priority of selected test case.

1. Repeat for i=1 to the number of test cases.
2. Repeat for j=1 to the number of test cases.
3. Initialize array T[i][j]
4. Repeat for i=1 to number of test cases
5. Repeat for j=1 to number of test cases
6. Store line numbers of line of source code covered by each test case
7. Repeat for i=1 to number of test cases
8. Store total number of lines of code
9. Repeat for i=1 to number of test cases
10. Initialize array nf[i] to zero
11. Set pos[i] =i
12. Repeat for i=1 to number of test cases
13. Repeat for j=1 to number of test cases
14. Initialize l to zero
15. Repeat for j=1 to length of the test cases
16. If pt[i] != 1
17. Increment nf[i] by 1
18. Increment l by 1
19. Repeat for i=1 to number of test cases
20. Repeat for j=1 to number of test cases
21. If nf[i]>0 then
22. u= nf[i]
23. nf[i]=nf[j]
24. nf[j]=u
25. u=pos[i]
26. pos[i]=pos[j]
27. pos[j]=t
28. Repeat for i=1 to number of test case
29. If nf[i]=1
30. Then increment count
31. If count=0 then goto end
32. Initialize priority[pos[0]]=m+1
33. Repeat for i=1 to length of test cases
34. Repeat for j =1 to lines of source code
35. If t1[pos[0][i]]=scode[j]
36. Scode[j]=0

In this paper we implemented, optimization algorithm in MATLAB R2009a to generate minimum number of test cases. Here the problems are coded and taken as m-file and by using genetic algorithm toolbox to generate test cases which helps to obtain test cases which are far more reliable [8]. GA requires number of variables which are needed to be set. Population size has great effect on the GA speed to obtain an optimum solution. Size

of population is typically between 20 and 100.Mutation and crossover operator helps to generate a solution which is in local optima. Later prioritization algorithm is implemented in C to indicate the order with which a test case may be addressed. A test case with highest rank has highest priority and the test case with second highest rank has second highest priority and so on.
For implementing the optimization and prioritization algorithm presented, we selected a problem i.e. to determine the smallest of 5 numbers. Test cases generation may depend on the parameter settings with number of variables taken as 5, population type is chosen to double vector, initial range is chosen arbitrarily randomly as [0,10], fitness scaling was based on rank, Selection is based on roulette wheel, crossover probability set to 0.8 by applying all types of crossover and mutation operators and selecting the one which gives the best results or best fitness value.

VII. RESULTS AND DISCUSSIONS

A code coverage technique [12] has been developed which is based on version specific test case prioritization and selects T' from T which is a subset of T. The technique also prioritizes test cases of T' and recommends use of high priority test cases first and the low priority test cases in descending order till time and resources are available. Here the prioritization is focussed around the changes in the modified program where all modified lines of source code are executed with minimum number of selected test cases. Researchers[12][13] developed a technique using deletion algorithm to generate minimum set of test cases and then used modification algorithm to prioritize those generated test cases which is basically based on the modified lines of source code. The deletion portion of the technique is used to update the execution history of test cases by removing deleted lines of source code and to identify and those test cases that cover only those lines which are covered by other test cases. The results generated by the researchers are shown in figure 1 and figure 2 which firstly optimize the test cases using deletion algorithm and then prioritizes using modification algorithm.

After generation of test cases deletion algorithm was applied to the set of set test cases. Thus, minimum number of test cases was obtained shown in figure 1.

```
2 3 4 5 8 16 20
Enter the length of test case 3
10
Enter the value of test case
6 8 9 10 11 12 13 14 17 18
Enter the length of test case 4
6
Enter the value of test case
1 2 5 8 17 19
Enter the length of test case 5
6
Enter the value of test case
1 2 6 8 9 13

 Enter the deleted lines of code 3
4 7 15
Test Case Exuction History after deletion :
T1      1520
T2      23581620
T3      68910111213141718
T4      12581719
T5      1268913
Remove Test Case 1
Remove Test Case 5
```

Figure 1: Test cases generated after applying deletion algorithm

After the implementation of deletion algorithm, modification algorithm was applied to prioritize the test cases to indicate the order with which the test cases may be addressed which is shown in figure 2.

```
Enter the length of Test Case 3:6
Enter the values of test case 1
2 3 5 8 16 20
Enter the values of test case 2
6 8 9 10 11 12 13 14 17 18
Enter the values of test case 3
1 2 5 8 17 19
Enter number of modified lines of code4
Enter the lines of code modified:6 13 17 20

Testcase        Matches
2               3
1               1
3               1modified array20
Testcase        Matches
1               1
2               0
3               0modified array
Test Case Selected.....

T1      Priority2

T2      Priority1
```

Figure 2.Prioritized test cases

For generating test cases, a model has been designed which accepts 3 integers as inputs and generates one output as root of a quadratic equation shown in figure 3.Firstly, the test cases were generated using decision coverage metrics but the result was not reliable as it gave redundant test cases shown in figure 4

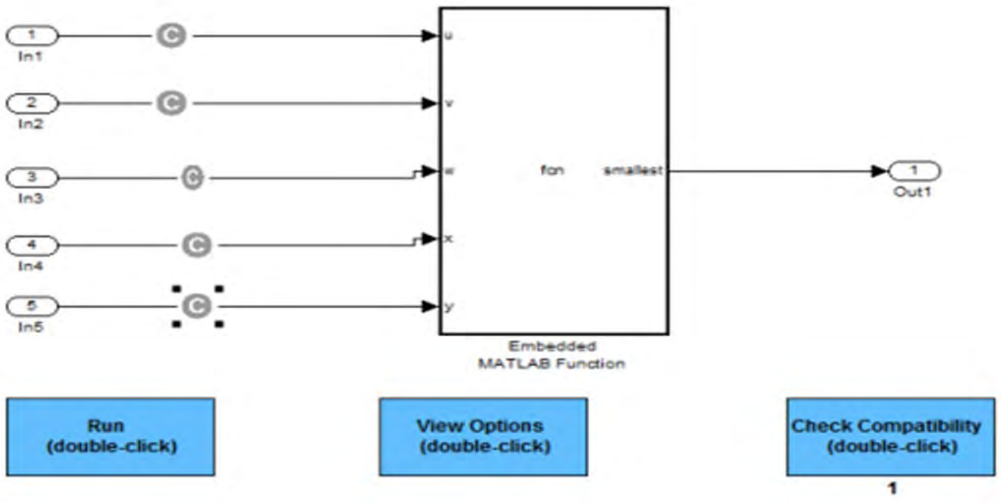

Figure 3: Model generating smallest of 5 numbers

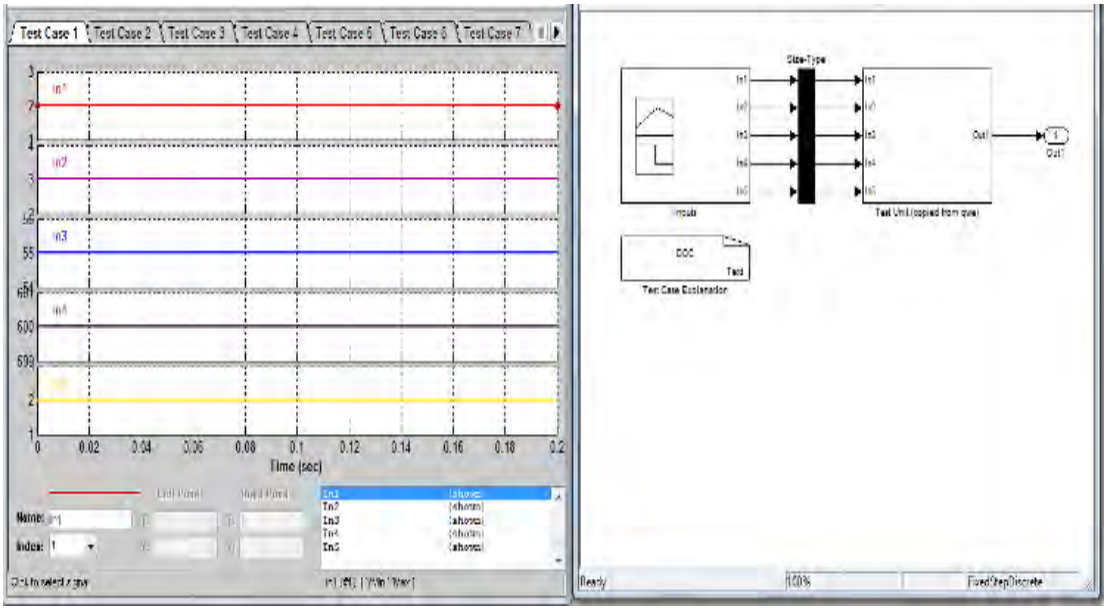

Figure 4: Test cases generated using decision coverage metric

So in order to obtain reliable results, we applied GA to generate optimized set of test cases till the specified conditions are specified which are shown in the table.This shows the optimized set of test cases for the given problem.

Table 1: Generated optimized set of test cases

	Input 1	Input 2	Input 3	Input 4	Input 5
Test Case 1	43.823	-1842.6	1.681	25.905	15.586
Test case 2	-153.677	22.683	-4.616	10.054	12.618
Test case 3	0.948	1.534	-1.31	-153.81	-20.321
Test case 4	14.915	30.186	-14.007	4.33	-157.01

Figure 5 and figure 6 shows the test case prioritization which shows that Test case 2 has highest rank so it holds highest priority; test case 5 has second highest priority so it holds second highest priority and Test case 3 has third highest priority.

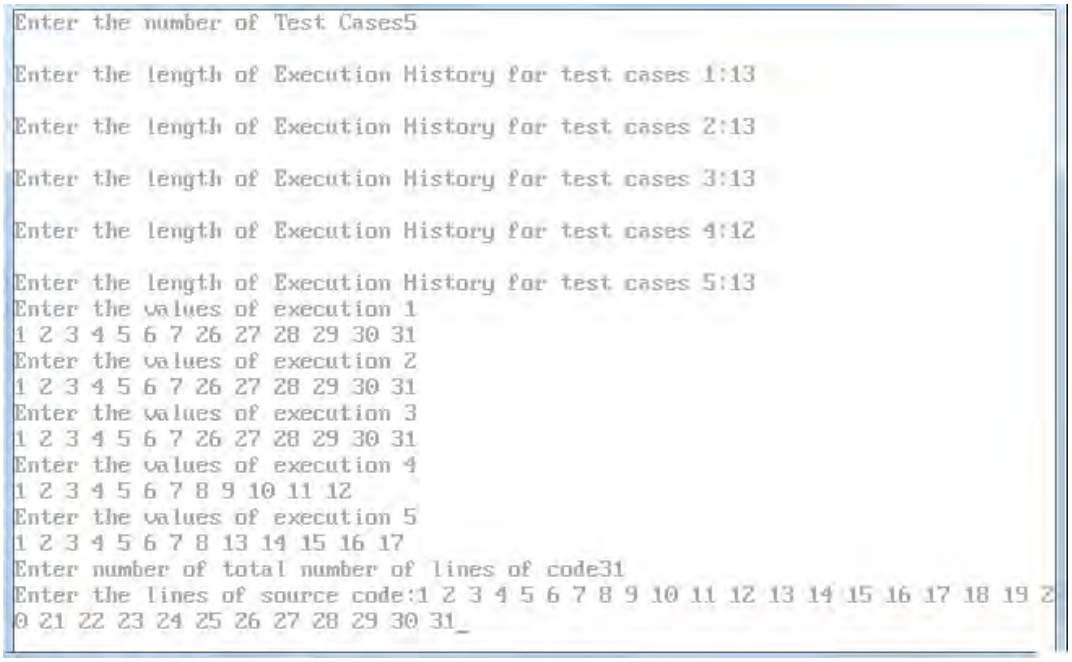

Figure 5: Prioritization of test case using proposed algorithm

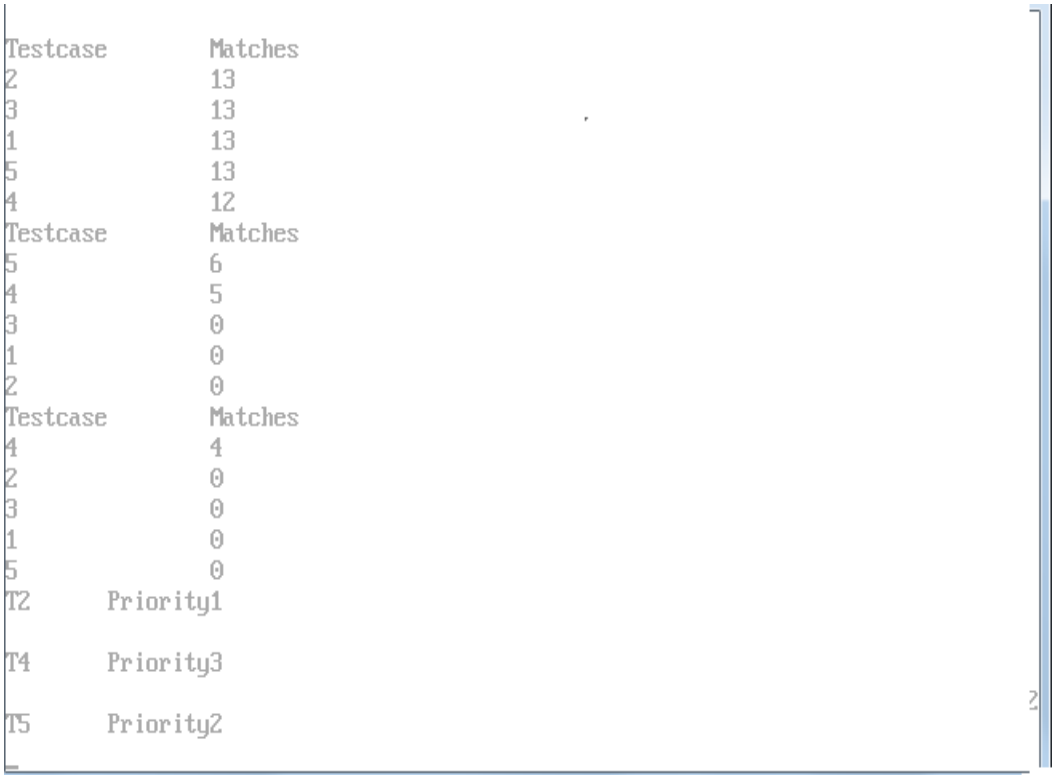

Figure 6: Prioritization of test case using proposed algorithm

VIII. CONCLUSION

In this paper, we proposed new algorithm for prioritization of test cases in software testing. In this algorithm firstly, we generated optimized set of test cases using genetic algorithm. Then we are applying the proposed algorithm on the set of optimized test cases directly. Thus the results obtained by researchers are focussed around the changes in modified program which is generally used in regression testing and we have applied the proposed algorithm on the whole model of smallest number system where the results are based on the whole model, the prioritization is done by including all the lines of source code. Thus, results are obtained by consuming fewer resources.

IX. REFERENCES

[1] Xanthakis S ,Ellis C, Skourlas C, Le Gall A, Kastiskas, Karapoulious K., "Application Of Genetic Algorithm To Software Testing", 5th international conference on software Engineering and its Applications,1992, pp 625-636.

[2] Roper M, MacleanI, Brooks A, MillerJ, Wood M., "Genetic Algorithm And The Automatic Generation Of Test Data", Technical Report RR/95/195, Department of Computer Science, University of Strathclyde 1995

[3] Jones B F, Sthamer H, Eyres D E. "Automatic Structural Testing Using Genetic Algorithm", Software Engineering Research Journal 1998; 41(2), pp 98-107

[4] Sthamer H. H., "The Automatic Generation Of Test Data Using Genetic Algorithm", Ph.D. Thesis, University of Glamorgan, Pontyprid, Wales, Great Britain 1995

[5] Pargas R P, Harrold M J, Peck R R, "Test Data Generation Using Genetic Algorithm", Journal of Software Testing, Verifications and Reliabilit 1999.

[6] Bueno P M S, Jino M, "Automatic Test Data Generation For Program Paths Using Genetic Algorithms", International Journal of Software Engineering and Knowledge Engineering Software Engineering (SBES) 2011.

[7] Michael C C, McGraw G E, Schatz M A, "Generating Software Test by Evolution", IEEE Transactions on Software Engineering, Vol. 27 (12), 2001.

[8] Praveen Ranjan Srivastava, Priyanka Gupta, Yogita Arrawatia, Suman Yadav. "Use Of Genetic Algorithm in Generation of Feasible Test Data", ACM SIGSOFT Software Engineering Notes, Vol. 34 Number 2, 2009

[9] Roya A lavi, Shahriar Lofti., "The New Approach For Software Testing Using A Genetic Algorithm Based On Clustering Initial Test Instances", International Conference on Computer and Software Modeling IPCSIT 2011 Vol. 14

[10] Pei M, Goodman E D, Gao Z, Zhong K., "Automated Software Test Data Generation Using A Genetic Algorithm" Technical Report GARAGe of Michigan State University June 1994.

[11] Watkins A. "A Tool For The Automatic Generation Of Test Data Using Genetic Algorithms", In Proceedings of the fourth Software Quality Conference, Dundee, Great Britain, 1995, pp. 300-309.

[12] Arvinder Kaur, "Development of Testing for Good Quality Object Oriented Software", Ph.D. dissertation, Guru Gobind Singh Indrapastha University, Delhi, India, 2006

[13] K.K Aggarwal, Yogesh Singh, Arvinder Kaur, "Code Coverage Based Techniques for Prioritization of Test Cases For Regression Testing", ACM SIGSOFT Vol. 29(05) September 2004