# A NOVEL APPROACH OF TEST CASE SELECTION IN REGRESSION TESTING BASED ON GENETIC ALGORITHMS

**Garima**[*]

**Mr. Deepak Kr. Singh**[**]

**Dr.Ajit Singh**[***]

**Abstract**

Regression Testing is a testing technique to test the modify software to ensure that changes are correct and do not adversely affect other parts of the software. Because the modifications to software may break functionality that used to work correctly, hence the regression test suite is large and needs an intelligent method to choose those test cases which will reduce the overall cost. In this situation test case selection techniques aims to improve the efficiency of regression testing. Many existing selection technique arrange the test cases on the basis of code coverage with respect to older version of the modified software. In this proposed approach, we identify the paths for test case execution and apply the elitist version of Genetic Algorithm. The overall aim of this research is to reduce the number of test cases that need to be run after changes have been made.

**Keywords**

Regression Testing, Genetic Algorithm, Fitness Function, Mutation, Crossover, Test Case Selection, Minimization and Prioritization.

[*] Sr. Lecturer, SIT, Mathura

[**] Asso. Prof. SIT, Mathura

[***] Prof. KEC Dawarahat

## I. Introduction

Regression testing is one noteworthy testing method that involves repeatedly running a test suite whenever the program is under test or the program environment is changed. Regression testing[1][2] is a testing activity that is performed to provide the confidence that changes do not harm the existing behavior of the software. Regression testing is performed between two different versions of software in order to provide confidence that the newly introduced features of the software under test do not interfere with existing features. Hence test suite tends to grow in size as software evolves, often making it too costly to execute entire test suites. There are number of different approaches have been studied to maximize the value of accrued test suites. Some of these approaches are:

1. Retest all: - It is a conventional method in which all the test cases are executed at once. Retest all method is not feasible and consumes more efforts.

2. Test suite minimization: - This method eliminates redundant test cases in order to reduce the number of test cases.

3. Test case selection: - It identifies the test cases that are relevant to some set of recent changes.

4. Test case prioritization: - This technique orders the test cases in such a way that early fault detection is maximized.

The proposed approach implemented a new regression test suite selection algorithm that select the test cases using Genetic Algorithm with the goal of minimizing the number of test cases that are likely to be found during time constrained execution.

This paper is organized as follows: section I gives the introduction of the regression testing and its techniques. Section II is helpful to understand the background of related work. Section III explains about the genetic algorithm including its operators and fitness function. Section IV gives the proposed approach, algorithm and results. Section V and VI concludes the paper followed by references

## II. Related Work

In this section, we give the brief overview of the previously proposed methods to find the optimal test sequence dynamically while performing the regression testing. Many researchers proposed various techniques on regression test suite reduction, prioritization and regression test case selection for improving the cost effectiveness of the regression testing.

Rothermal and Harrold give a technique for regression test selection. These techniques construct control flow graphs for a procedure or module and its modified version and use these control flow graphs to select tests that execute changed code from the original test suite[3]. Saifur-Rehman Khan, Aamer Nadeem proposed novel test case reduction techniques called TestFilter that uses the statement-coverage criterion for reduction of test cases[4].

Ananda and Kiran[5] proposed an approach for reducing the cost of regression testing in black box level. This approach applied in real time case studies and estimated the regression cost using cost estimation model. This approached is worked in three phases. Phase 1: "Reduced Test Suite" is derived by applying proposed approach on the original test suite[6]. Phase 2: "Reduced Regression Test Suite" is derived by applying a regression test selection (from phase 1). And phase 3: a testing cost estimation model is applied on the reduced regression test suite and empirically calculated the regression testing cost reduction.

Kaushik et.al.[7] proposed a paradigm called dynamic prioritization which involves changing the order of test cases during the testing process. Kaur et.al[8] proposed a new Genetic Algorithm to prioritize the regression test suite is introduced that will prioritize test cases on the basis on complete code coverage. Jaiswal proposed a method for test case selection in regression testing using genetic algorithm.

The Genetic Algorithm would also automate the process of test case prioritization. Rothermel et.al[9] describe several techniques using genetic algorithm for list execution information to prioritize test cases for regression testing.

Kumar et.al[10] presents a combined approach by which the stated problems are resolved in efficient manner. Suman et.al[11] proposed a method to reduced the time and cost of regression testing using Genetic Algorithm as well as it provide simple testing flow as the test cases will be minimized. Designed system used the partially mapped crossover and swap mutation to optimize the test sequence.

Giries et.al[12] has proposed a structural oriented automatic test data generation technique that uses genetic Algorithm guided by data flow dependencies in the program to full fill the all user criterion. The program to be tested is converted into control flow graph where each node represents a block in a program and the control flow of the statements. The test case generation by this proposed genetic algorithm is more effective as compared to the random testing technique.

Conrad et.al in 2010[13] proposed a test case prioritization technique in regression testing using genetic algorithm. The paper presented a wide verity of mutation, crossover, selection and transformation operator that were used to reorder the test suite. Genetic algorithm yielded timer results as compare to other techniques.

R.Krishnamoorthi et.al [15] proposed a prioritization technique based on both testing time and potential fault detection information to intelligently reorder a test suite using genetic algorithm in regression testing.

Arvinder Kaur et.al [14] proposed a method for prioritizing and automatic test case generation in regression testing using genetic algorithm, which is based on the two fitness function criterion- 1. The maximum fault covered in minimum execution time and 2. Total code coverage. These fitness functions helped in selecting suitable population for problem.

In the proposed approach, the genetic algorithm is apply on selected the path from the code coverage and reduce the time and cost of regression testing along with it will provide simple testing flow as the test cases will be minimized.

## III. Genetic Algorithm

Genetic algorithms [16] are the heuristic search and optimization techniques that mimic the process of natural evolution. Genetic algorithm is a method that imitates the genetic and evolutionary mechanisms, which some of the similar behavior of genes, introduction such as crossover recombination, mutation, selection, and elimination of algorithm and other improvements into process.

The most commonly used genetic algorithm operators are summarized as follows:

1. **Selection:** - Selection operator, also known as Reproduction operator. It selects individual from the population from the population by a probability. roulette wheel selection strategy, in which parents are selected according to their fitness values. In this strategy, every individual of the population will receive a fitness function value calculate APBC and then we can calculate the ratio of the fitness value and the sum of all individual fitness value.

$$\text{Ratio}(i) = f(i) / F(1) + F(2) + \ldots\ldots + F(s)$$

Where F(i) is the fitness value of individual I and S indicates the total number of individuals in the population. The sum of ratio (i) in this population equal to 1. After arrange individuals of the population from low to high according to their fitness values, generate a random number r € [0,1].

2. **Crossover: -** The role of crossover operator from the middle generation is to randomly select two individuals, crossover their parts of gene and to generate two new individuals with a crossover strategy. System generates a random number between 0 and 1, if the random number is less than the crossover probability $P_{c,}$ and than does the cross operation. The process of parent individual m, n cross generation offspring of individual p, q is as follows:

- Generate a random crossover point k, k is bigger than 1, less than n (n is the number of test sequence in the test case).
- Copy the first k test cases of m into p.
- Remove k test cases in n and then copy of rest into p.
- Similar to generate p, individual q consists of first k test cases in the n and n-k test cases m which is removed of the k test cases.

3. **Mutation: -** Mutation is the occasional introduction of new features in to the solution strings of the population pool to maintain diversity in the population. Mutation operator changes a 1 or 0 or vice versa with a mutation probability of. Mutation is an important way to maintain species diversity in genetic algorithm. The role of mutation operator is randomly selected genes, and changes its value with a probability to form a new individual. Mutation operation process is as follows:

- Generate a random between 0 and 1, if the random number is less than mutation probability $P_{m,}$ and then do the mutation operation.
- Randomly selected two cases in the test sequences, and to exchange its location.

**Fitness Function: -** A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution. The input of test case prioritization based on genetic algorithm is a sort of test cases; the output is the test case sequence. Therefore, the real numbers can be used to represent test cases. With assign a real number for each test case, test sequences can be decomposed of an array with these numbers. Individual fitness value reflects the adaptability of the individual in solution space. The fitness value determines the individual is to multiply or die. Generally, the fitness value is 0 and 1. The larger the value is, the more appropriate the individual is. And it is copied into the next generation more probably. The test case coverage information generally can be used to quantify the possibility of finding errors. Than construct the fitness function with average block coverage. The formula is given by:

$$APBC= 1- ((TB_1 + TB_2 +TB_3 +\ldots..+TBm)/nm) +1/2n)$$

The optimization problems are solved by genetic algorithms recombination and replacement operator where recombination operator is frequently used where as the other operator is optimal and applied for solving optimization of problem[17]

## IV. Proposed Approach and Validation: -

Here we are concerned to select the test case in regression testing with the generation of testing with the generation of test data that can achieve a threshold level of coverage (statement coverage and path coverage) using genetic algorithm.

**Problem Statement:** Our aim is to select a test suite that covers the maximum no. of lines executed of the program. For this purpose we use the proposed Genetic algorithm approach.

We generate initial population by randomly generating test cases and then execute the path under test for each member of the initial population. With the help of proposed genetic algorithm approach we tried to calculate fitness for each member of initial population. For calculation the fitness function we used formula:

$$\text{Fitness} = (\text{total statement} - \text{uncovered statement}) / \text{total statements}$$

**Algorithm**

```
Begins
1. Generate initial population by randomly generating test cases.
2. Execute path under test for each member of initial population.
3. Compute fitness for each member of initial population.
4. while (particular level of coverage is not achieved and max
      number of iterations not reached )
5. Select some members having high fitness value from current
      population.
6. Generate new population from selected members of current
      population using crossover and mutation.
7. Execute path under test for each member of new population.
8. Compute fitness for each member of new population.
9. End while
```

```
10. Return new population

      End
```

We used this algorithm to generate efficient test data which is based on the chosen path.

For the validation of this approach we use the triangle problem. In triangle problem three parameters a, b, and c are passed as input to the program and based on these values result is generated that the triangle is equilateral, isosceles, scalene not a triangle or the input belongs to invalid range. We calculated the number of lines executed as per each test case (input values) with the help of gcov coverage tool. Gcov is a tool we can use in conjunction with GCC to test code coverage in our programs.

The triangle problem is described as follows:

```
#include<stdio.h>
int main() {
int a, b, c, validinput=0;
printf("Enter the side 'a' value:\n");
scanf("%d", &a);
printf("Enter the side 'b' value:\n");
scanf("%d", &b);
printf("Enter the side 'c' value:\n");
scanf("%d", &c);
if((a>0)&&(a<=100)&&(b>0)&&(b<=100)&&(c<0)&&(c<=100)) {
if(((a+b)>c)&&((c+a)>b)&&((b+c)>a)) {
validinput= 1; }}
else{
validinput=1; }
if(validinput=1){
if((a==b)&&(b==c)){
printf("The triangle is equiletral \n"); }
else if ((a==b)||(b==c)||(c= =a)){
printf("The triangle is isoscale\n"); }
else{
printf("The triangle is scalene \n"); }}
else if(validinput==0){
printf("The values do not constitute the triangle \n"); }
else{
printf("The input belongs to invalid range\n");
return 0; }
}
```

| Test Case | a | b | c | Expected Output | % of lines executed |
|---|---|---|---|---|---|
| 1 | 0 | 6 | 6 | Invalid input | 62.07 |
| 2 | 1 | 6 | 6 | Not a triangle | 58.62 |
| 3 | 4 | 6 | 6 | Isosceles | 68.97 |
| 4 | 6 | 6 | 6 | Equilateral | 65.52 |
| 5 | 3 | 4 | 5 | Scalene | 63.54 |
| 6 | 3 | 3 | 5 | Isosceles | 68.97 |
| 7 | 4 | 3 | 4 | Isosceles | 68.97 |

Table 1.1: Number of Lines Executed As Per Test Case

| Generation | Average Fitness value |
|---|---|
| 1 | 72.4931032 |
| 2 | 75.7385608 |
| 3 | 81.2845082 |
| 4 | 86.9876548 |
| 5 | 89.0569271 |
| 6 | 92.8743617 |
| 7 | 93.7931032 |
| 8 | 94.3276439 |
| 9 | 95.1724136 |
| 10 | 95.8721890 |

Table 1.2: Average Fitness as Per Generation

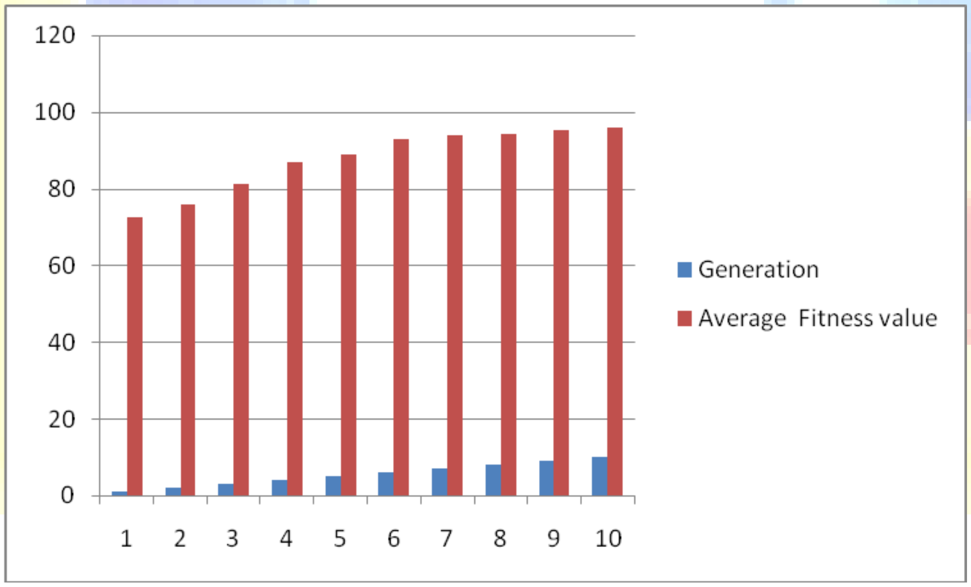

Fig 5.1: Generation vs. Average Fitness

As shown above table 5.2, describe about the test case and its expected result as well as the percentage of the no. of lines executed of the program. Using this statement coverage analysis value we calculated the average fitness value which is shown in table 5.2 that describe about maximum coverage of the program with respect to each new test case combination. Using table 5.2 value we plotted a graph between average fitness value and each generation, showing maximum coverage of the program with new test case combination.

## V. Conclusion :-

In this paper, we proposed a new test case selection approach in regression testing by selecting the statement coverage of a program and select the path among the set of paths in an order to achieve the testing objective. We can use this approach to execute the test case on the selected paths. By the use of statement coverage path selection strategy, infeasible paths are also identified which can reduce the effort, time, and cost. This strategy also eases the process of regression testing without affecting the quality of software testing. In addition to path selection efficient test cases are generated by elitist genetic algorithm to cover the selected path. This combined approach will be very helpful to the software tester in order to generate a test set to cover the desired path.

## VI. References: -

[1] Elmar Juergens, Benjamin Nummel, Florian Deissenboeck, Martin Feilkas, Christian Schlgel Andreas Wobeke, "Regression Test Selection in Manual System Tests in practice" In 15th European Conference on Software Maintenance and Reengineering, pages 301-312, 2011.

[2] Gregory M. Kapfhammer "Empirically Evolution Regression Testing Techniques: Challenges, Solutions and a Potential way forward". In 4th International Conference on System Testing Verification and Validation workshop, pages 99-102, 201.

[3] G.Rothermal and M.J.Harrold, "A safe, efficient regression test selection technique", ACM Transactions on Software Engineering Meth. 6(2): 173-210, April.

[4] Saif-ur-Rehman Khan Nadeem, A. Awais, "TestFilter: A Statement Coverage Based Test Case Reduction Technique", IEEE Multi topic Conference, page: 275-280, Dec 2006.

[5] A. Ananda Rao, Kiran Kumar, "An Approach to Cost Effective Regression Testing in Black Box Testing Environment", IJCSI, Vol.8, Issue 3, No, 1, May 2011.

[6] Kiran Kumar J, A. Ananda Rao, M.GopiChand, K.Narendra Reddy, "An Approach to Test Case Design for Cost Effective Software Testing", IMECS-IAENG-2009.

[7] N.Kaushik, M. Salehie, L.Tahvildari, S. Li, M. Moore(2011) "Dynamic prioritization in regression testing" IEEE fourth international conference on software testing, verification and validation workshops, pp: 135-138.

[8] A. Kaur, S. Goyal, "A Genetic Algorithm for Regression Test Case Prioritization using Code Coverage", Internationl journal on computer science and engineering, vol. 3, pp: 1839-1847.

[9] G.Rothermal, R.H.Untch, C.Chu, M.J.Harrold(2001) " Prioritization test cases for regression testing" IEEE transcation on software engineering (to appear), PP: 1-33.

[10] A.Kumar, S. Tiwari, K.K.Mishra, A.K.Misra(2010)" Generation of efficient test data using path selection strategy with elitist GA in regression testing". IEEE, pp: 389-393.

[11] Suman, Seema, " A genetic algorithm for regression test sequence optimization", International Journel of Advance Research in computer and communication engineering, vol. 1, issue 7, sept.2012.

[12] Girgis, "Automatic test generation for data flow testing using a genetic algorithm", Journal of computer secience, 11(6), 2005, pp.898-915.

[13] A.P.Conrad. R, S.Roos, " Empirically Studying the role of selection operators during search based test suite prioritization", In the Proceedings of the ACM SIGEVO Genetic and Evolutionary Computation Conference, Portland, Oregon, 2010.

[14] Arvinder Kaur, Shubhara Goyal, "A genetic algorithm for regression test case prioritization using code coverage, International Journal on computer science and engineering (IJCSE).

[15] R. Krishnamoorthi and S.A.Sahaaya, Arul Mary, "Regression test suite prioritization using genetic algorithms", International Journal of Hybrid Information Technology, vol. 2, No.3, July, 2009.

[16] Letica M.Press et.al. "Path Selection in the structure Testing: Proposition, implementation and application of Strategies". In XXI International Conference of the Chilean Computer Science Society, Nov, 2001.

[17] K. R. Walcott, "Prioritizing regression test suites for time-constrained execution using a genetic algorithm" [online] available at www.cs.virginia.edu/~krw7c/gaprioritizations.pdf.