ISSN: 2277-9655
Impact Factor: 4.116

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

AN APPROACH TO GENERATE TEST CASES AUTOMATICALLY USING GENETIC ALGORITHM

Deepika Sharma*, Dr. Sanjay Tyagi

* Research Scholar, Department of Computer Science and Applications, Kurukshetra University, Kurukshetra-136119

Assistant Professor, Department of Computer Science and Applications, Kurukshetra University, Kurukshetra-136119

DOI: 10.5281/zenodo.55533

ABSTRACT

Software testing is a very crucial part among all phases of software life cycle model in software engineering, which leads to better software quality and reliability. The main issue of software testing is the incompleteness of testing due to the vast amount of possible test cases which increase the effort and cost of the software. So generating adequate test cases will help to reduce the effort and cost of the software. The purpose of this research paper is to automatically generate test cases using the concept of boundary value analysis and genetic algorithm and then results are compared with random testing which shows better quality of the software and outperforms random testing.

KEYWORDS: Boundary value analysis, Genetic Algorithm, Random Testing, Software testing, Test case generation.

INTRODUCTION

Software testing provides surety of software quality, reliability and acceptability by discovering errors. Software testing can account for more than half of the cost of a software product. As the main purpose is to reduce the excessive cost of the software as well as time, the testing activity should incorporate an effective and efficient method with the highest possible level of automation. For testing software performance or quality, test data or test cases must be needed. So test data generation is the important and first step for software testing. Functional and structural testing are the two fundamental techniques used to identify test cases [1]. But these two approaches are very time consuming. So there must be some mechanism for reducing the testing effort. Here one of the mechanisms that will be discussed is by generating the test cases automatically. For automatic test case generation, genetic algorithm is being used for implementation using the concept of boundary value analysis in this paper. With the help of genetic algorithm we can generate suitable test cases easily. Holland proposed GA as a heuristic approach on the basis of "Survival of the fittest" [2]. Genetic algorithm is a simple approach for search and optimization related problems. The process of genetic algorithm starts with an initial population, followed by various genetic operators such as selection, crossover, mutation and replacement.

Genetic algorithm reaches stopping conditions when the population converges towards an optimal solution [3].

PROBLEM FORMULATION

After the source code has been generated, and then it must be tested to uncover as many errors as possible before delivering it to the customer. For this, we require a series of test cases that have an assurance of finding errors. Test case helps the user to test all possible combinations in the application. Because complete testing is usually impossible as the domain of possible inputs of a program is too large to be completely used in testing a system. Therefore, testing can be a sampling method. And an appropriate sample must be selected so that it contains error sensitive data for software testing. If test data chosen are irrelevant then the probability of finding faults in the system are less. Testing can be performed either manually or automatically but manual testing take lot of effort and time [4]. So automatic

testing is the best option to choose for testing the software. Various techniques have been proposed for generating test cases automatically such as random test data generator, path oriented test data generator, goal oriented test data generator, and intelligent test data generator[5]. Genetic algorithms is a useful tool for search and optimization related issues [6].

To automate the software test, the evolutionary algorithm must itself be an optimization task. Now, the first step is to randomly generate an initial population of test data set. Now test is executed on each chromosome of the population which represents test data. Next, the fitness value of each individual is evaluated and then the individuals with higher fitness value are selected to generate new offspring using different combination methods from it. The main criteria of the evolutionary algorithm was to select the best solution of test data set from the population and remove the test data having less fitness value or apply different combination method to generate offspring test data from the selected individuals so that offspring have best combination of selected individuals that will be able to fulfill various test objectives. A new population will be generated that will replace the old population and contain better individuals to form an optimal solution. After that the process performs the iterations, starting from selection until the optimal solution is found or any stopping condition is reached.

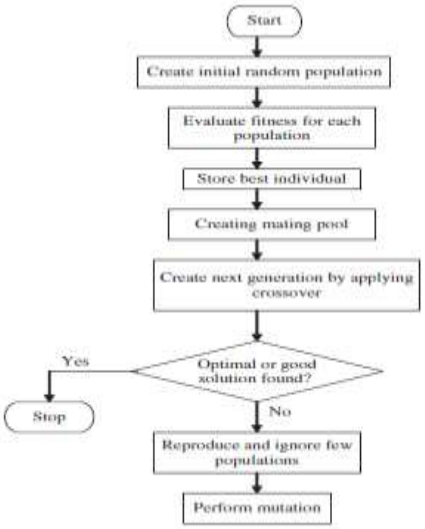

Figure1: Flow Chart of Genetic Algorithm[3]

RELATED WORK

Random Testing is the easiest approach for test case generation. Random testing generates test data for any type of program, as it is independently generated from their operational profile. It is not able to cover all types of faults since random generation of test case may not execute some statements having faults. The probability of finding at least one error in software by random testing depends only on the number of test points [7].

Random Testing chooses test data set randomly from uniform distribution and perform testing using these test data set. Random testing program is viewed as a worst case of program testing [8] because of its inability to find failure in the system. A mixed final testing was recommended [8], that merges random testing with value based testing.

The easiest and cheaper technique for test data generation is the random testing which require less effort and time for test data generation [9].

A path and branch based approach to fitness computation for program test data generation using genetic algorithm has been discussed by ankur pancholi [10]. Here, a new approach is used to significantly improve the search performance of the system using two step processes. In the first step, target node sequence was computed and then compares it with actual path to compute fitness function.

A program that exercises a selected branch in a program to generate test data has been presented by mathur [11]. Here a new input is derived from the initial input through any of the paths from the selected branch. This method dynamically switches among the paths that reach the branch by refining the input.

A model based testing has been discussed by briand [12]. It allows an automatic test case generation at a very high level. Here, the authors proposed a technique whose function uses state machine and genetic algorithm based selection algorithm.

A paper on automatic test suite generation using genetic algorithm has been presented by girdhar gopal [1] which used an evolutionary approach for test case generation. Here, an algorithm was proposed to increase the efficiency and effectiveness of the software based system.

A paper on "An approach to generate software test data for a specific path automatically with genetic algorithm" focused on the reliability of the software and ensuring the quality by automating test data generation for path testing [13]. Here, a new fitness function to evaluate the individuals of population and drive GA for a specific path has been presented to search the appropriate solutions.

RESEARCH METHODOLOGY

It is observed that most of the errors occur at the boundaries of the system and not at the middle, because there is always some uncertainties at the boundaries. So the most important factor for automated test case generation is the boundary value analysis. Boundary value analysis is a technique of selecting test cases at the boundaries of the system. For example, input values for two variables that may have any value from 100 to 300 are generated as: (200, 100), (200, 101), (200, 200), (200, 299), (200, 300), (100, 200), (101, 200), (299, 200). The input domain is shown in fig 2. Here, each dot represents a test case and inner rectangle is the boundary or domain of the inputs. Any point within the rectangle is considered as the reasonable point of the function.

Boundary value analysis identifies various test cases by developing five values, minimum value, just above the minimum value, nominal value, just below the maximum value and the maximum value[1].

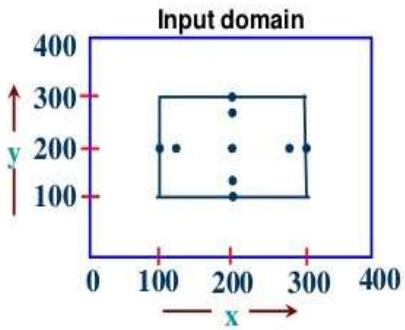

Figure 2: BVA for function of two variables x and y with boundaries [100, 300]

In this paper, we carried out the identification of test cases at the boundaries automatically through genetic algorithm and random testing and then compared the results of both. Genetic algorithm and random testing both start with the random initial population and then GA select the optimum population using the fitness of the individuals whereas random testing works randomly throughout the process. For this experiment, the distance from boundaries specifies the fitness of individuals. The algorithms are coded in MATLAB R2009b.

Genetic algorithm for test suite generation

The proposed genetic algorithm for test suite generation has been presented here. The major methods of GA are discussed and then the results are presented.

Initial population

The proposed GA uses the real values for the input variables of the program. Initial population is generated randomly in their specific range. The length of chromosome will be specified based on the number of variables. Ideally, the initial population should be taken as large as possible in order to explore the whole search [3]. So for the search space, initial population is taken in 5 less than the lower bound of variable and 5 more than the upper bound of a variable. For example, if lower bound and upper bound of variable are 10, 20 respectively, and then initial population is generated in the range from 5 to 25.

Fitness Function

Fitness function is determined by calculating the difference of individuals from the lower and upper bound of variables. A variable which is closer to the boundary considered fit.

Selection

The main purpose of selection process is to select individuals from the population having highest fitness values [3]. So, here the algorithm selects effective members from the current population that will be further mated to form new and better offspring from them. In the selection method the GA uses *Tournament selection method*, whereas random testing chooses two parents randomly from the population using *random testing method*. These two methods are described below:

Tournament Selection: For the selection of new population, tournament strategy provides selection by holding a tournament competition among various individuals. Such a tournament selection is constructed as follows:

```
for i=1:2
  tournament_size=randi(pop_size);
  temp=fit(1);
        for j=1:tournament_size
            if(temp<=fit(j))
                temp=fit(j);
                index(i)=j;
            end
        end
end
j=1;
for i=1:c_size
    parent1(i)=curpop(index(j),i);
    parent2(i)=curpop(index(j+1),i);
end
```

Where curpop(i,j) denotes data members of the current population. Here, tournament is performed among various individual and chooses the two parents on the basis of their fitness values as described above.

Random selection: This technique randomly selects parents, so that every individual get an equal chance of being selected for mating.

```
For i=1:pop_size
        j=rand(l);% l=effective members of population
        Select chromosome ci from the effective  members of population
End;
```

Crossover

During crossover, two parents selected from selection process to interchange data or information at a random position in the chromosome to produce two new offspring. Here, the aim of crossover is to find new population by selecting the best individual and then swapped them to create better offspring, so that the population is enriched with better individuals. Crossover will be performed based on some crossover probability. The crossover probability is a particular number, which specifies the chances of crossover operation to be performed. Arithmetic crossover is used

for each pair of selected parents with crossover probability of 0.7. It produces two offspring's which are linear combinations of their parents as follows:

```
alpha=0.5;
 for i=1:c_size
  offspring1(i)=alpha*parent1(i)+(1-alpha)*parent2(i);   offspring2(i)=alpha*parent2(i)+(1-
 alpha)*parent1(i);
end
```

Where alpha is the weight, which is the dominant individual during reproduction and lies between 0-1.

Mutation

In mutation process a single bit is flipped in the chromosome to generate new offspring[3]. The mutation is performed if crossover exploits the current solution to find better individual then the mutation will help in it but the probability of performing mutation is always kept small. Here uniform mutation is performed with mutation probability of 0.1 as follows:

```
aa=rand();
 if(aa<0.1)
      index=randi(pop_size);
     offspring=pop_array(index,1);
      mutate=offspring-0.01;
 end;
```

Where pop_array is the population array from which an offspring is selected randomly for mutation.

Replacement

Here entire population of genomes is replaced at each generation. It means newly generated offspring's would move forward to the next generation and replace all the chromosomes of the current generation.

EXPERIMENTAL RESULTS & DISCUSSIONS

All input variables are taken from the user, so that testing can be done easily with all the parameters considered. User interface in MATLAB is as follows:

Input:- No. of variables: 2, Population size: 15, No. of generations: 100

Limit of first variable- lower bound: 5, Upper bound: 15

Limit of second variable -lower bound: 10, Upper bound: 20

Output:-With Genetic algorithm, test cases generated are as follows:

Table1: Test Cases after 100 generations (GA)

Variables	Variable1	Variable2
1	5.72	9.60
2	10.67	9.81
3	12.31	12.59
4	13.20	11.24
5	14.30	15.54
6	16.34	19.69

For random testing, test cases generated are as follows:

Table 2: Test Cases after 100 generations (Random)

Variables	Variable1	Variable2
1	15.89	10.38
2	6.85	5.46
3	8.07	20.01
4	16.04	21.59
5	2.74	7.27
6	7.23	12.43

ISSN: 2277-9655
Impact Factor: 4.116

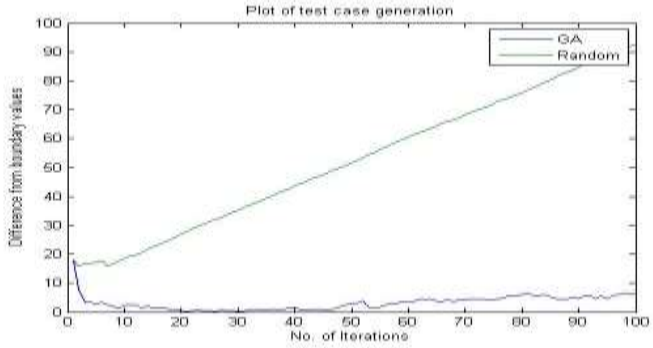

Figure 3: Difference from boundary values for first entry from table1 & table2 for 100 no. of generations.

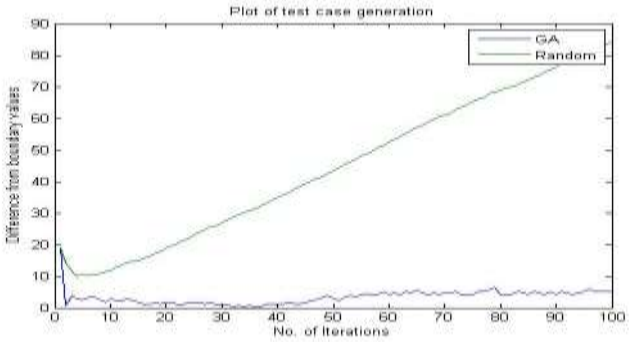

Figure 4: Difference from boundary values for second entry from table1 & table2 for 100 no. of generations.

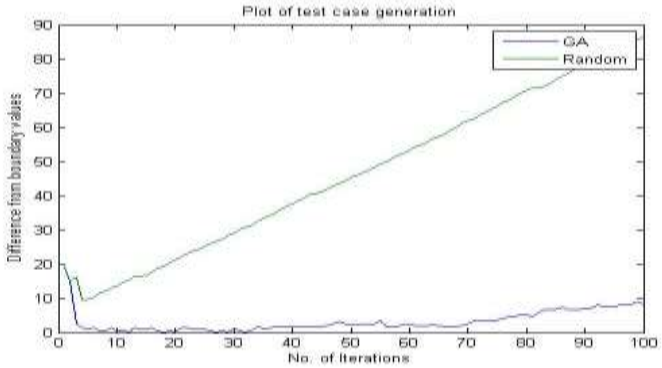

Figure 5: Difference from boundary values for third entry from table1 & table2 for 100 no. of generations.

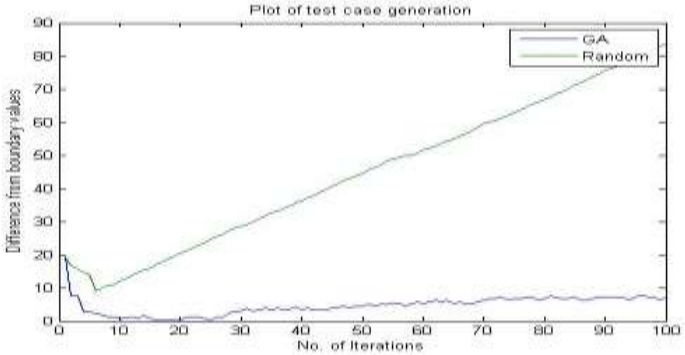

Figure 6: Difference from boundary values for fourth entry from table1 & table2 for 100 no. of generations.

ISSN: 2277-9655
Impact Factor: 4.116

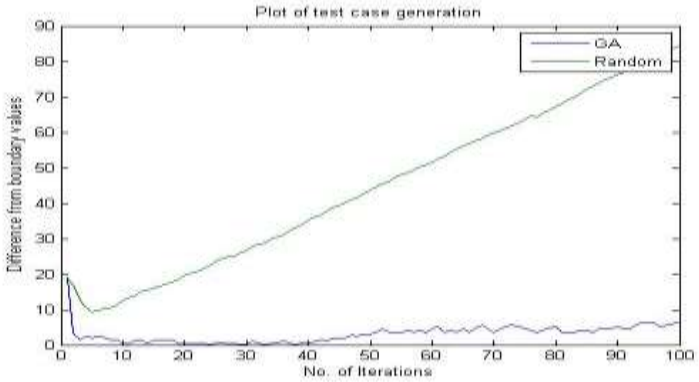

Figure 7: Difference from boundary values for fifth entry from table1 & table2 for 100 no. of generations.

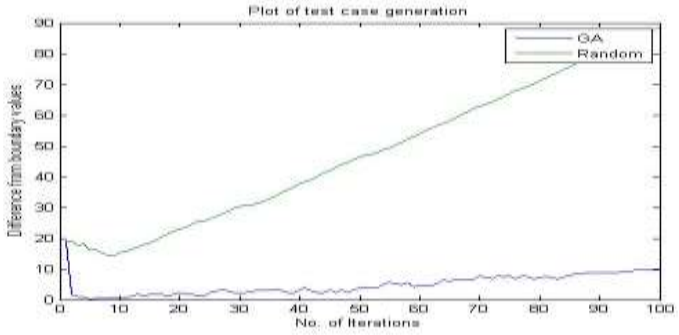

Figure 8: Difference from boundary values for sixth entry from table1 & table2 for 100 no. of generations.
In this experiment, a simple GA is applied, for the pop_size of 15. While probability of crossover is set to be 0.7 & mutation 0.1. Selection process used is tournament selection for 100 numbers of generations.

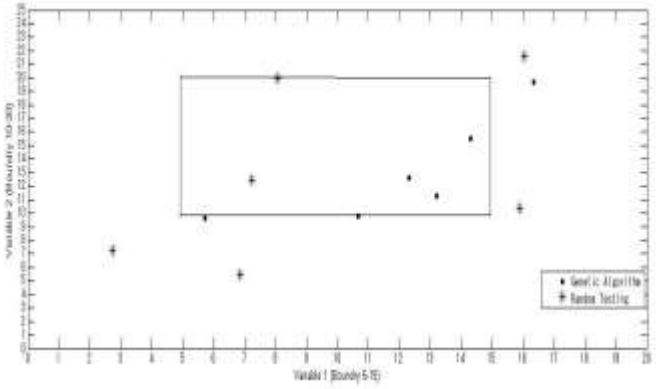

Figure 9: Results after final test case analysis

It is clearly visible from the experiment shown in fig 9 that test cases with GA are spread within the boundaries of the variables whereas random testing generate most of the test cases outside of the boundary.
Another run is carried out with following inputs:-
Input:-
No. of variables: 2, Population size: 20, No. of generations: 500
Limit of first variable- lower bound: 10, Upper bound: 30
Limit of second variable- lower bound: 20, Upper bound: 40
Output:-With Genetic algorithm, test cases generated are as follows:

ISSN: 2277-9655
Impact Factor: 4.116

Table 3: Test Cases after 500 generations (GA)

Variables	Variable1	Variable2
1	13.01	19.60
2	10.67	20.81
3	23.26	21.28
4	24.59	25.41
5	29.99	29.95
6	30.75	35.10

For random testing, test cases generated are as follows:

Table 4: Test Cases after 500 generations (Random)

Variables	Variable1	Variable2
1	13.15	15.17
2	25.11	19.17
3	27.09	21.88
4	25.48	44.39
5	16.27	16.83
6	12.73	28.52

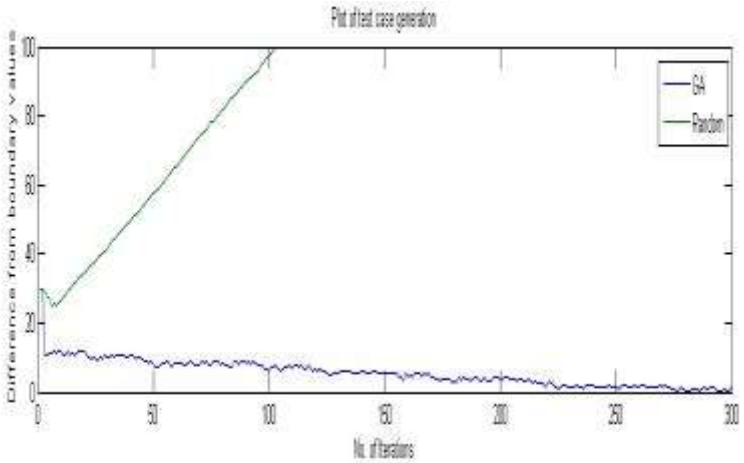

Figure 10: Difference from boundary values for first entry of table3 & table4 for 500 no. of generations.

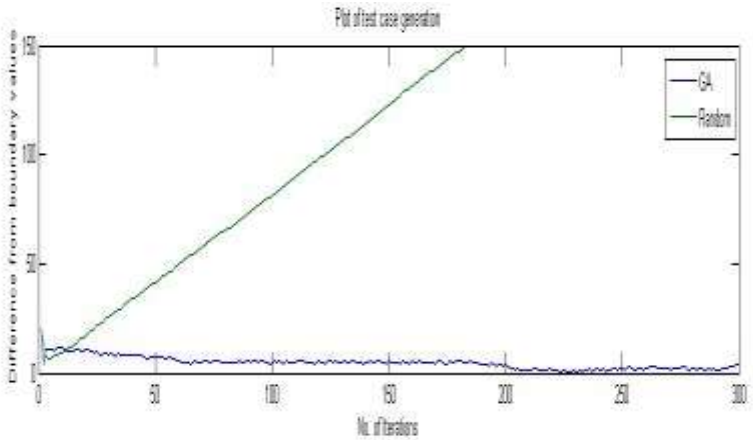

Figure 11: Difference from boundary values for second entry from table3 & table4 for 500 no. of generations

ISSN: 2277-9655
Impact Factor: 4.116

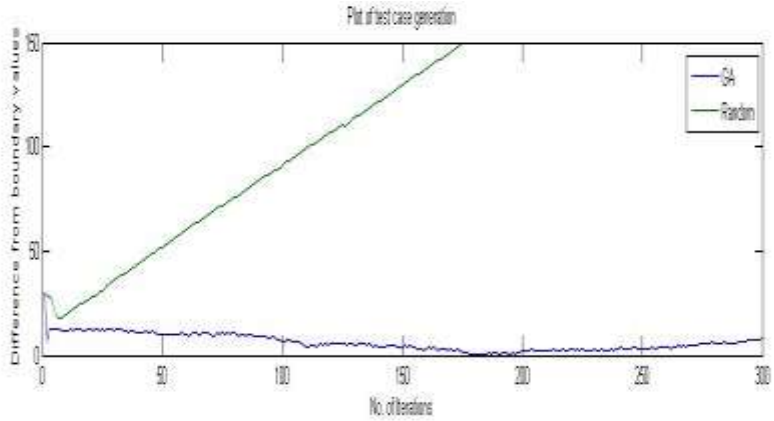

Figure 12: Difference from boundary values for third entry from table3 & table4 for 500 no. of generations.

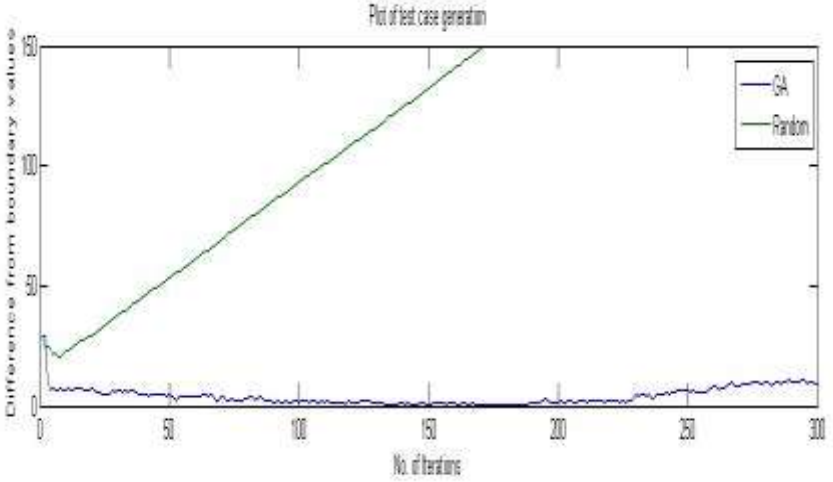

Figure 13: Difference from boundary values for fourth entry from table3 & table4 for 500 no. of generations.

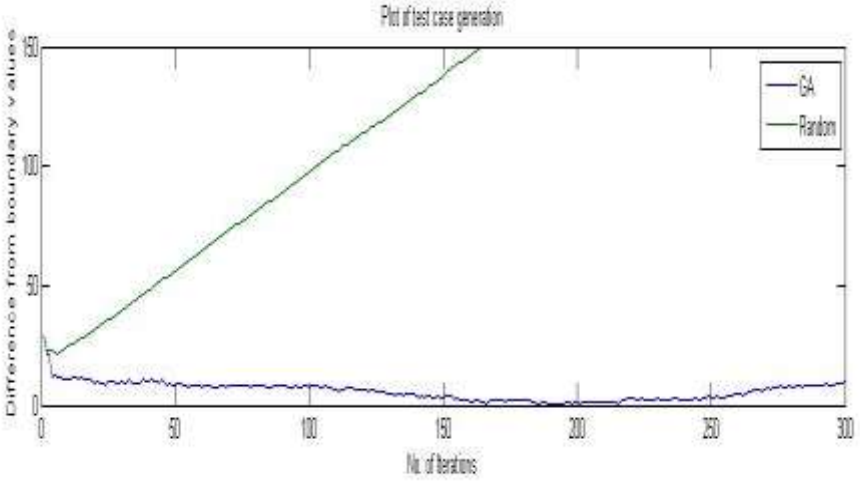

Figure 14: Difference from boundary values for fifth entry from table3 & table4 for 500 no. of generations.

ISSN: 2277-9655
Impact Factor: 4.116

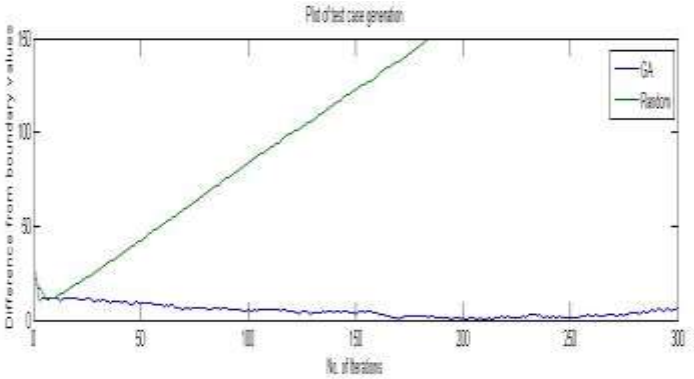

Figure 15: Difference from boundary values for sixth entry from table3 & table4 for 500 no. of generations.

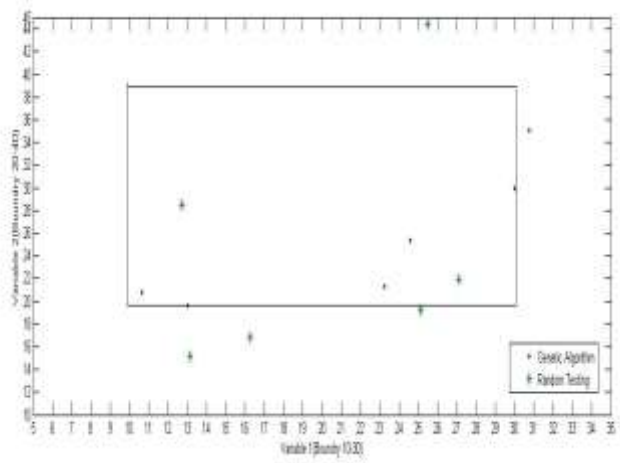

Figure 16: Results after final test case analysis

Here, the number of generation taken is 500, pop_size is set to 20 and then experiment is performed. It is clearly visible from the experiment in fig 16 that test cases with GA are spread within the boundaries of the variables whereas random testing generates most of the test cases outside of the boundary. It also shows that as the number of generations increase, the results will get better because on each iteration, the better individuals are chosen and then swapped to get enriched with better population of individuals.

CONCLUSION & FUTURE SCOPE

To increase the efficiency and quality of the testing process, the automated test case generator is required. Here genetic algorithm is used for automatic generation of test case using the concept of boundary value analysis. The overall quality of the testing is enhanced as compared to the random testing. The results shown specify a promising approach for automated test case generation using boundary value analysis concept. The future scope of test case generator can go further using different types and parameters of crossover (blend crossover, single point, uniform, et al.), mutation (basic bit, uniform, et al.) and selection (rank et al.). Here, boundary value analysis is used for an automated test case generator, but other additional fields such as control flow graphs, path testing, regression testing etc. can also be used for test case generation.

REFERENCES

[1] Girdhar Gopal, Surjeet Singh, and Rakesh Kumar, "Automatic Test case Generation using genetic algorithm," International Journal of Scientific & Engineering research, pp. 1135-1141, 2013.
[2] Holland J., Adaptation in natural and artificial systems. Ann Arbor: University of MIchigan Press, 1975.

[3] S.N Deepa S.N Sivanandam, Introduction to gentic algorithms.: Springer, 2008.
[4] Bichitra Kalita Hitesh Tahbildar, "Automated Software Test Data Generation:Direction of Research," International Journal of Computer Science & Engineering Survey, pp. 99-120, Feb 2011.
[5] Neelabh Sao Neema Patel, "Optimal Test Data Generation Using Hybrid Techniques IWD & ACO," Journal of Emerging Technologies and innovative Research, pp. 91-98, Oct 2015.
[6] Rijwan Khan and Mohd Amjad, "A review on automated test case generation using genetic algorithms," International Journal of Advanced Research in Computer Science and Software engineering, Dec 2014
[7] Richard Hamlet, "Random testing,".
[8] S.C.Ntafos J.W.Duran, "An Evolution of random testing," IEEE transaction on software testing, pp. 438-444, July 1984.
[9] D.C Ince, "The automatic generation of test data," The computer journal, pp. 63-69, 1987.
[10] Gurusaran, Gaurav Mishra Ankur Panchauri, "A path and branch based approach to fitness computation for program test data generation using genetic algorithm," in 1st international conference on futuristic trend in computational analysis and knowledge management, 2015, pp. 49-55.
[11] Aditya P.Mathur,Mary lou Soffa Neelam Gupta, "Generating test data for branch coverage," IEEE, pp. 219-227, 2000.
[12] Lionel Briand, Andea Arcuri,Shaukat Ali Hadi Hemmati, "An enhanced test case selection approach for model based testing," in FSE-18, New Mexico,USA, Nov 2010.
[13] Chunhua HU,Luming LI Yang CAO, "An Approach to generate software test data for a specific path automatically with genetic algorithm," IEEE, pp. 888-892, 2009.