International Journal of Scientific Engineering and Science
Volume 1, Issue 12, pp. 69-73, 2018. ISSN (Online): 2456-7361

Software Test Case Optimization Using Genetic Algorithm

Sandeep Goyal, Pranmohan Mishra, Amrit Lamichhane, Dr. Parul Gandhi
Faculty of Computer Applications, Manav Rachna International Institute of Research and Studies, Faridabad

***Abstract*—** *Testing is work escalated and tedious process. Testing can be completed in two ways either manually or automatically. Tester conducts the testing process with the assistance of different accessible automated testing tools and strategies. Testing software is essentially the way toward recognizing an arrangement of information which fulfils the criteria set for testing. Test data generation is the way of collecting the data which meet the testing criterion. Parts of research have been finished by numerous specialists and they created many test data generator e.g. random test data generators, symbolic test data generators and dynamic test. This paper presents the streamlining investigation of the experiment age in light of the Genetic Algorithm and produces test cases which are significantly more reliable. This paper stresses on generation of test case using genetic algorithm. Genetic algorithm is the adaptation technology of their own. The purpose of the research paper is to implement the genetic algorithms to reduce the test cases and reduce cost, time and effort to give good quality software.*

***Keywords*—** *Software testing, genetic algorithm, test data, selection, mutation, crossover.*

I. INTRODUCTION

Several approaches were proposed for the test case generation, mainly random, path-oriented, goal oriented and some intelligent approach. Random techniques find the test cases based on assumption made about the fault distribution [1]. Path-oriented technique is usually used to control the flow of information so that the cases of testing of these paths can be covered and can be generated. These techniques have been further classified as static and dynamic. Static techniques are often based on

Symbolic execution, while dynamic technique requires data for executing the program under test. Goal-oriented techniques cover the cases of test covering a selected goal such as a statement, branch or path taken. Intelligent techniques or automated test depends on complex computes to test case generation cases. The objective of the test is to discover the errors in the software and the process executed with the intention of finding errors. If the errors are not recognized by the test, it means that test case set is not adequate. Activities are carried out to evaluate and verify properties of a program to check whether it meets the necessary results or not. Detection of failure is a challenging task using limited resources. It is easy to recognize the mistakes in the software because it is said that the software is not continuous, therefore test the boundary values in the Border Value Analysis or by using the criteria such as Path coverage and are not adequate to ensure correctness and exhausting testing is ineffective [5]. The programs are more complicated due to the dynamic nature, if a failure occurs during the initial test and the code changes, the behavior of the software which was previously passed, is not guaranteed at pre-error testing cases. Therefore the test should restart. Software is now being used in critical situations where failure is only inconvenient, from the perspective of software development organization, resulting in defects products result in loss of goodwill. Thus, the only option to do this is the first time before product is shipped to customer.

In this paper, we present the results of our research into the application of GA search approach, to make the hand wet on genetic algorithm by solving general numerical problem. The paper is structured in the following way: section IV describe basic structure of genetic algorithm, in section V, VI we discussed use of proposed algorithm in different testing methodology and in section VII describe use of GA in numerical domain.

II. SIGNIFICANCE OF AUTOMATED TESTING

Software systems should be reliable, available, and secure and to secure these objectives techniques are being used to avoid mistake, fault tolerance, defect, and theft of mistake etc. Fault detection is an integral part of the testing. it is usually done for the following purposes: (a) Quality assurance, (b) for verification and verification (V & V): Test V & V process is used as a tool. Tester can claim based on the tester explanations the result of the test is whether the product works under some conditions or not. Test with the purpose of verifying product tasks is called clean test. The deficiencies are that they can only validate that software works for specified test cases. A finite number of tests cannot be verified that the software works for all situations. On the contrary, only one failed test is enough to show that software does not work to see dirty test tests. The goal of dirty test is breaking down the software and software must possess the adequate handling capabilities to avoid a significant level of dirty tests. For Reliability Assessment: Software Reliability [10] has important connection with many aspects of the software.

The structure, and the amount of testing is based on an operating profile (Estimate of relative frequency of different usage Input) for the program, tests can serve as one of the statistical sampling method for failure data for reliability estimation. That is why the test is important activity in software development but this is a time consumption process and sometimes consuming more over 50% of the total efforts required for development. The cost of software testing can be very low If the test process is automated

International Journal of Scientific Engineering and Science
Volume 1, Issue 12, pp. 69-73, 2018. ISSN (Online): 2456-7361

III. Need for GA in Software Testing

The Demerits of Manual Testing-

Operation speed is limited because it is done by humans.
Cost, high investment in case of time
Limited availability of resources
Duplicacy in test cases
Check out unskilled and wrong examination

Merits of Genetic Algorithms in Software Testing-

Parallelism is an important feature of genetic testing
During operation, there is less chance of trapping in the extreme code of a test because it is running in a search space.

With the same encoding, only the problem needs to be changed according to change.

IV. Genetic Algorithm

GAs was created by John Holland and his understudies and partners at the College of Michigan [2]. Genetic Algorithm replicates the process of evolution to take care of the issue of software test case optimization. GA run all the more productively for the not having any positive strategy and restricted time stamp.GA may not be the best technique for any task yet strong and pretty much appropriate for more intricate optimization task. It is particularly most appropriate for high power computerized computing. The calculation starts with a pool or populace of conceivable answer for a given issue these arrangements at that point experience change like in nature for delivering new kids or arrangement. The process is iterated over different ages. The chromosome (i.e. the GUI control in our application) is assigned to a particular value called (fitness value). The chromosomes with higher fitness values are given extreme priority to mate to deliver the fitter chromosome or streamline arrangement. In this way we continues adjusting, altering and choosing others to the process until the point when we achieve the final solution. It is regularly used to discover ideal solution or close ideal answer for the complex issue which generally would take the long time surpassing the assessed time period. GA gets better solution when we have the large search space i.e. noisy environment and applicable in various domains such as control, robotics, signal processing, game playing, scheduling, design etc. It is valuable when we have expansive search space (substantial no of solution) including extensive number of parameter. Conventional strategies are too expensive regarding processing time. Genetic Algorithm is one of the cases of counterfeit wise calculation that tries to locate the best answer for a particular problem domain. Manual testing is too moderate and costly on account of human association for age of experiments. So it was a sole requirement for any product improvement association to discover ideal method for producing the experiments. Therefore automated testing techniques were presented, for example, Genetic Algorithm, Ant bee colony optimization, swarm optimization, these algorithm produces the test case consequently guaranteeing that the experiment are not repetitive, utilizing the ideal time and spending plan.

4.1 Algorithmic Paradigm used for Genetic Algorithm

Firstly, we select the parents out of initial population for mating. Mutation and crossover operation is carried out among two parents to get new offspring with the help of mutation or cross over operator [3]. The new offspring replace the individual population in the sample and repeats until the final solution is reached.

Genetic algorithm follows the following steps to solve any problem.
Step 1. Initialize the sample population consisting finite no. of chromosome
Step 2. Calculate the fitness value (adaptability to a problem) of each chromosome using objective function (constraints to reach the optimum solution)
Step 3. Select the fitter chromosome for further iteration
Step 4. Crossover
Step 5. Mutation
Step 6.Repeat the step 2 to 5 until the final criterion is met or optimal solution is obtained
Step 7. Best solution is found.

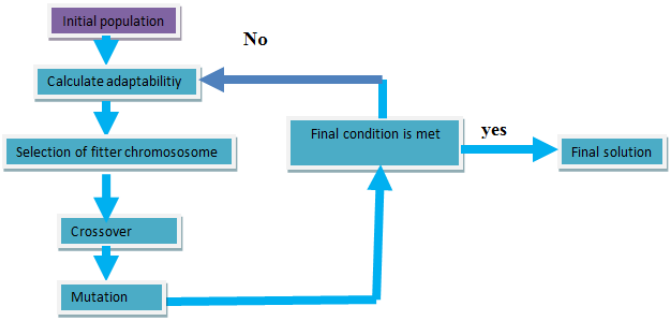

Fig. 1. Diagrammatic representation of genetic algorithm.

4.2 Process of Genetic Algorithm

The first decision that must be taken before executing the genetic algorithm is representation of solution [4]. The representations which are not good to the solution drive the GA to poor execution. In this way we have to pick proper representation of solution regarding issue having appropriate mapping capacity amongst genotype and phenotype.

4.2.1 Population initialization

Populations are initialized using these techniques. In random initialization, we take initial population as the completely random solutions. In Heuristic initialization we take the initial population based on the known heuristic techniques for problem.

It has been observed that the initial population taken on the basis of heuristic technique result in population having same solution with less diversity. Thus experiment conducted with random sample lead to the most optimal solution.

4.2.2 Fitness function calculation

Fitness function basically computes of likelihood of specific being chosen out of population. It analyses the degree of fitness of a particular solution with respect to problem in consideration. GA repeatedly calculates the fitness of a solution in multiple iteration. Therefore it should be sufficiently fast otherwise it makes the testing process exceptionally slow

4.2.3 Parent determination

Parent determination is the significant movement in the GA as choice of good parent for mating to create new offspring drive to better solution. Be that as it may, it is not considered effective to select the greatly fit arrangement as it will prompt the arrangements which are near each other in this way prompting the loss of decent variety.

4.2.4 Crossover

Crossover is the process of producing a new offspring by taking more than one parent. New child inherits the genetic material or feature of parents indulging in crossover process. One point cross over random point is taken and tails of two parents are swapped to get new offspring. Multipoint crossover two alternating segment of parents are swapped to get new offspring. Uniform crossover, it is based on gene level cross over from their parent chromosome rather than segment level as in one point and two point crossover. Mixing ratio is 0.5, thus each child contains the half of the genes from first parent and remaining from second parent although points can be randomly chosen.

4.2.5 Mutation

Mutation is defined as slight change in chromosome to get a new solution. It alters a one or more genes from initial chromosome to find new solution. The resultant solution can be entirely different from previous one or may be identical to some extent. It helps to maintain diversity in solution and is applied with low probability.

It is based on the concept of exploring the search domain. It has been concluded that it converge the solution to desired one while crossover not.

4.2.6 Stopping condition

Termination conditions in GA specify the stopping criteria after the desired solution is obtained in few numbers of iterations. Termination condition in GA can occur because of the following reasons:

a) Finite number of generation
b) Estimated time frame is completed
c) Estimated budget and allocated resource is reached
d) Most optimized solution is obtained
e) Iteration are not producing better result

V. White Box Testing Using Genetic Algorithms

White box testing checks the internal structure of the program. The tester works with code, loop and condition statement. In some research works, code coverage and data flow testing are discussed using the genetic algorithm.

5.1 Path Testing

Ranjan Shrivastav and Tai-Hun Kim [6] worked on the technique to test data generation using genetic algorithms. This weighted CFG uses the path to find all possible paths in the program to find test errors. If the program does not have any of the loops, then there is an infinite number of paths in it. Here, a large number of test cases need to be included in every way. This is the reason that NP can be a complete problem by covering all possible paths in the test. Our algorithm works on the control flow graph (CFG). CFG operates on the independent route for the new set of statements or conditions. During the testing, every independent path should come at least once. Keshavraj and Raza Javidan, worked on path tests using genetic algorithms. In the Path Testing, we deal with it to draw program control flow graphs. During the test, we will have to consider the following parameters for the performance of the path-

1. Need to cover each independent route.
2. The time of the investigation is not more than the scheduled time.

The main purpose is to test the path using the genetic algorithm software quality control. Poonam Saini, Sanjay Tyagi [7], worked on test data generation in path tests using genetic algorithms. In software testing, testing cases or testing data is a more important aspect to manually test effectively. The time consuming process of test data generation is compared to the automatic test date generation. Due to the effort and time-consuming process, software testing is a customization problem, therefore, the optimization technique is used to create the appropriate data for the genetic algorithm, the methods were covered in each every independent path in control flow graph of control flow graph.

5.2 Data Flow Testing

Mohaheb R. Chiragis [8] worked on automated test data generation on data flow testing using genetic algorithms. Genetic algorithm accepts the defuse path of the program for testing. Defuse path defines the number of input variables, and their definitions, and the use of each input variable as well; it analyzes the population size, crossover and the possibility of mutation. Here, find the number of test cases for a defuse path to perform data flow testing. Algorithm has an integer vector that records the difference in each defuse path. The initial value of the integer vector is zero. It uses calculations or as a counter to check the effectiveness of the current population

VI. Black Box Testing Using Genetic Algorithms

Black Box testing, this tests the software and software performance, software specifications and user requirements. In some research, the functional testing and regression testing are discussed using genetic algorithms.

6.1 Functional Testing

Shay Eyal, and Abraham Kandel [9] worked on the implementation of evolutionary techniques to improve the effectiveness of the test cases. The cases of "poor" testing should be abolished and cases of "good" testing are evaluated to test. In functional testing, we check the functionality of the software generated in the genetic algorithm regarding randomization cases and numerical values in relation to the population. Then find the values of the fitness function to find the number of errors to test the cases. "Good" populations act as test cases while others end. In this way, test sets regard the set of cases as priority. In this way population or test case is generated in few number of iteration. Consequently, functional testing using genetic algorithm expected to reveal the fault exposing test cases.

International Journal of Scientific Engineering and Science
Volume 1, Issue 12, pp. 69-73, 2018. ISSN (Online): 2456-7361

VII. Generation of Test Data Using Genetic Algorithmic Approach

Here is the simple example of genetic algorithm to maximize the following function
$F(x) = x^3$ over the range of integer 0 to 15.
We are representing x with four bit unsigned binary bits because binary string of four bit can represent 16 number and GA work well with binary representation
F(x) is the function which calculates the fitness of each individual
Firstly we take the four randomly generated solutions as:
1001,0110,1010,0100

For evaluating the fitness of each solution, first we decode the binary representation into integer as:

1001 ⟶ 9, 1010 ⟶ 10, 0100 ⟶ 4, 0110 ⟶ 6

According to objective function $f(x) = x^3$

9 ⟶ 729, 6 ⟶ 216, 10 ⟶ 1000, 4 ⟶ 64

Sting no.	Initial population	X value	Fitness value	Probability	Expected count N*prob(i)
1	1001	9	729	0.36	1.44
2	1010	10	1000	0.50	2
3	0100	4	64	0.03	0.06
4	0110	6	216	0.11	0.44
Total			2009	1	3.94
Average			502.25	0.25	0.99
Maximum			1000	0.50	2

Thus string number 2 has maximum chance of selection.

We divide the range into 4 bins, sized according to relative fitness of solution.

String	Probability	Associate bin
1001	0.36	0…0.36
1010	0.50	0.36…0.86
0100	0.03	0.86…..0.89
0110	0.11	0.89……1

By generating four uniform random number between 0 to 1 we choose the string for next generation after observing which bin they fall into

Random	Bin	Chosen String
0.20	0…0.36	1001
0.40	0.36…0.86	1010
0.60	0.36…0.86	1010
0.93	0.89……1	0110

Random number generator determines the pair of string for us to mate.
For the first pair of string:
We select the cross over point after the third digit which yields new offspring as:

1001 ⇨ 1000

1010 1011

For second pair of string:
We select the crossover point after the first two bit in the string and yields new offspring as:

1010 ⇨ 0110

0110 1010

In next iteration we again calculate the fitness of population

Sting no.	Initial population	X value	Fitness value	Probability	Expected count N*prob(i)
1	1000	8	512	0.17	0.68
2	1011	11	1331	0.44	1.76
3	0110	6	216	0.07	0.28
4	1010	10	1000	0.33	1.32
Total			3059	1	1.04
Average			764.75	0.25	1.01
Maximum			1331	0.44	1.76

Total fitness has gone from 2009 to 3059 in a single generation. Hence we can find the optimized solution in few generations if we repeat the same process mentioned earlier. Because the algorithm has already come up with the solution i.e 1011(x=11) as possible solution.

VIII. Conclusion

Testing is the way toward approving the client requisite. It expends one third of the product development life cycle and moderately exorbitant and imperative than different process of software development life cycle. As, it is impossible to test the software extensively with manual testing, in this paper we have examined the how evolutionary process is useful in software testing. GA operates on large search space choosing the best out of numerous solutions. It also shows how GA adjusts them to any condition to think of better solution. GA can be connected to extensive number of problems like n-queens problem, travelling salesman problem etc. to generate optimal solution. GA has made major headways in the field of software testing for the age of experiments conceivable because of which we can find out the ideal solution which is exhibited by taking most modest number of framework in this paper for producing experiments. In this paper basic numerical problem is unraveled with. This shows how efficient it is for generation of test data in each generation. This will definitely pave the way the way for further work in this area.

Acknowledgement

We are highly indebted to Dr. Parul Gandhi and Dr. Prasenjit Banerjee for their constant support and guidance under which this work has been possible.

References

[1] Kulvinder Singh, Iqbal Kaur, and Rakesh Kumar, "Automatic test case generation using genetic algorithm with antirandom population," *International Journal of Advanced Computer Engineering*, vol. 5, issue 1, pp. 21-27, 2012.
[2] Fundamental of Genetic algorithm: RC Chakrabarty, www.myreaders.info
[3] Benjamin J. Lynch, *Optimizing with Genetics Algorithm*, 2006.

[4] https://www.tutorialspoint.com/genetic_algorithms/index.htm
[5] Software Engineering and Testing by K.K Agrawal and Nasib Singh
[6] Praveen Ranjan Srivastava and Tai-hoon Kim, "Application of genetic algorithm in software testing," *International Journal of Software Engineering and Its Applications*, vol. 3, issue 4, pp. 87-96, 2009.
[7] Poonam Saini and Sanjay Tyagi, "Test data generation for basis path testing using genetic algorithm and clonal selection algorithm," *International Journal of Science and Research (IJSR)*, vol. 3 issue 6, pp. 2319-7064, 2014.
[8] Moheb R. Girgis, "Automatic test data generation for data flow testing, using a genetic algorithm," *Journal of Universal Computer Science*, vol. 11, Issue 6, pp. 898-915, 2005
[9] Mark Last, Shay Eyal, and Abraham Kandel, "Effective Black-Box Testing with Genetic Algorithms," Department of Computer Science and Engineering, Ben-Gurion University of the Negev, BeerSheva, Israel, 2005.
[10] R. L. Michael, *Handbook of Software Reliability Engineering*, McGraw-Hill Publishing, ISBN 0-07-039400-8, 1995.