International Journal of Engineering Research & Management Technology

May- 2015 Volume 2, Issue-3

Email: editor@ijermt.org www.ijermt.org

"Shkills..... "Sharpening Skills..... Serving Nation"

Application of Software Testing using Genetic Algorithm

Mohit Saifi
M. Tech Scholar
Computer Science and Engineering
Al-Falah School of Engineering and Technology
Dhouj, Faridabad, Haryana

Shahid Sagar
Assistant Professor
Computer Science and Engineering
Al-Falah School of Engineering and Technology
Dhouj, Faridabad, Haryana

ABSTRACT:
Different types of software testing techniques and methods have been projected for taking care of these issues. Use of evolutionary algorithms for usual test generation has been an area of interest for many researchers. Genetic Algorithm (GA) is one such type of evolutionary algorithms. In this research paper, we current a study of Genetic Algorithm approach for addressing the different issues encountered during software testing. This paper presents a method for optimizing software testing effectiveness by identifying the mainly critical path clusters in a program. We are increasing a more selective approach to testing by focusing on those parts that are largely critical so that these paths can be tested initial. By identifying the large amount critical paths, the testing effectiveness can be improved.

KEYWORDS: genetic algorithm, Software testing techniques, Test data, Software application.

INTRODUCTION:
The software testing is most important part for on the increase high class software. Software testing is utilize to catch the faults, which is reason for software failure. Software testing is time intense and laborious work. It consumes fifty percent of software system growth resources. Proof is the process to make sure the item for utilization satisfies the condition imposed at the start of the increase phase. In other words, to make sure the product behaves the way we want it to. Support is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements. Testing can be done either manually or automatically by using testing tools. Automated.[1]software testing is much better from manual testing. Automated software testing reduces the cost of software development, testing results can be increased, and also the test runs would be fast. However, very few test data generation tools are available at this time. Various types of techniques have been proposed for generating test data or test cases automatically. [3]Evolutionary algorithm is a subset of evolutionary computation. Evolutionary algorithm often performs well approximating solutions to all type of problems. Evolutionary testing is an emerging methodology for automatically producing high quality test data. Genetic algorithm is one such evolutionary algorithm formulated by John Holland in United States during late sixties. Genetic algorithm has been applied in many other different types of problems for generating test plans for feasible test cases in different arias, functionality testing, and genetic algorithm has also been used model based test case generation. Genetic algorithm is used to generate test cases in object oriented unit testing as well as black box testing. In this research paper, we present the results of our research into the application of GA search approach, to identify the most error prone paths in a software construct, software testing techniques where GA is efficiently used is presented. This paper is divided into 5 sections. Section 2 describes briefly the working of a GA. Section 3 describes the initialization of GA with real world data. In section 4, we describe the testing techniques with GA and section 5 describes the conclusion part. [3]

GENETIC ALGORITHM:

In the field of Artificial Intelligence a genetic algorithm is a search heuristic that mimics the process of natural selection. Sometimes it is called Meta heuristic. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithm belongs to the larger class of evolutionary algorithm which generates solution to optimization problems using techniques inspired by natural evolution. Such as Inheritance, Mutation, Selection and crossover. [5][6]The possible solutions to problem being solved are represented by a population of chromosomes. A Chromosome can be a binary string or a more elaborate data structure that makes up a chromosome is called a gene. The initial pool of chromosomes can be randomly produced or manually created using processes such as greedy algorithm. The fitness function measures the suitability of a chromosome to meet a specified objective. The selection function decides which chromosomes will participate in the evolution stage of the genetic algorithm made up by the crossover and mutation operator. The crossover operator exchanges genes from two chromosomes and creates two new chromosomes. The mutation operator changes a gene in a chromosome and creates one new chromosome.

The pseudo code of a basic algorithm for genetic algorithm has well-defined steps:

Initialize (population)
Evaluate (population)
While (stopping condition not satisfied) do
{
Selection (population)
Crossover (population)
Mutate (population)
Evaluate (population)
}

The algorithm will iterate until the population has evolved to form a solution to the problem, or until a maximum number of iterations have taken place (suggesting that a solution is not going to be found given the resources available). Genetic algorithm uses three operators on its population which are described below.

A. SELECTION:

A selection scheme is applied to determine how individuals are chosen for mating based on their fitness values. First the fitness values can be defined as capability of an individual to survive and reproduce in an environment. It is calculated using fitness function proposed in the algorithm. Weights are used to find the relative contribution of a path to the fitness calculation. Thus, more weight is assigned to a path which is more "critical". Selection generates the new population from the old one, thus starting a new generation. Each chromosome is evaluated in present generation to determine its fitness value. This fitness value is used to select the better chromosomes from the population for the next generation. The fitness function are using here is

$$F = \sum_{i=1}^{n} wi$$

Where, wi = weight assigned to i-th edge on the path under consideration

The algorithm works by assigning weights to the edges (depicting flow) of Control Flow Graph on the basis of the importance of path in which the edge lies. Higher weights are assigned to the edges of path corresponding to the critical section of the code for example branch statements, loops, control statements etc. for which testing is necessary. After all the fitness function values are intended, the possibility of selection pj for each path j, so that

$$pj = Fj / \sum Fj$$

n= initial population size

$$ck = \sum_{j=1}^{k} pj$$

Then cumulative possibility ck is measured for each path k with equation.[7][8]

B. CROSSOVER OR REPRODUCTION (RECOMBINATION):

After selection, the crossover operation is applied to the selected chromosomes. It involves exchange of genes or sequence of bits in the string between two individuals. Crossover happens according to a crossover possibility pc, which is an adjustable parameter. For each parent selected, generate a random real number r in the range [0, 1]; if r < pc then select the parent for crossover. After that, the selected data are formatted randomly. Each pair of parents generates two new paths, called offspring.

The crossover technique used is one point crossover done at the midpoint of the input bit string. After crossover, the mutation operator is applied to a randomly selected subset of the population.[7]

C. MUTATION:

Mutation is performed on a bit-by-bit basis. Mutation alters chromosomes in small ways to introduce new good traits. It is applied to bring diversity in the population. Every bit of every chromosome in the offspring has an equal chance to mutate (change from '0' to '1' or from '1' to '0'), and the mutation occurs according to a mutation possibility pm, which is also an adjustable parameter. To perform mutation, for each chromosome in the offspring and for each bit within the chromosome, generate a random real number r in the range [0, 1]; if r < pm then mutate the bit.

I. INITIALIZATION OF G.A.:

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem. But typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found. Following examples of GA in real word.[9]

***A.* IN SUDOKU:**

Harder puzzle may require larger population sizes in order to reliably find solution. The solver may stick on an almost solution and struggle to make the final adjustment to complete the puzzle.

***B.* IN DESIGNING A NEW CAR:**

This correspond to taking the engine from successful car and breaks from another successful car. The breaks from one car have been used in the design of a new car.

II.GA IN SOFTWARE TESTING TECHNIQUES:

In this section we will discuss in detail the applications of GA in black box testing and white box testing.[10]

A. APPLICATIONS OF GA IN WHITE BOX TESTING:

White box or structural testing are based on the code itself. It can be done in the form of data flow testing or path testing. Here we taking the Path testing, it involves generating a set of paths that will cover every branch in the program and finding the set of test cases that will execute every path in this set of program path.

Dr. Velur has been proposed the approach for test cases generation from directed graph. A directed graph is represented by G = [V, E], where V represents state or vertices and edges represents flow of control. Thereafter, a graph containing n nodes is represented by an incidence matrix of order n * n where an entry '1' in the matrix represents edge between nodes and '0' represents no edge or connection between them (see Fig 1). By using the nodes of graph as the base population, pair of nodes are generated which are selected as parents by applying the dual graph generation technique proposed as shown in Fig 2.

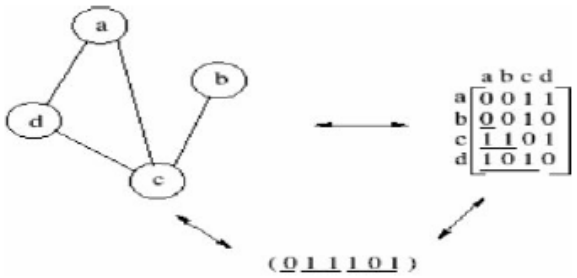

Fig. 1 Graph representation as a binary string.

Fitness is calculated by using the 'Current maximum cliquealgorithm'and Approximation algorithm'. Fitness is assigned by finding the clique of size 5 and the number of independent sets of size 5 in the population which comprise of number of graphs in the population. The graph with 0 fitness value indicates the clique of size 0 and no independent sets of size 5 in the graph.

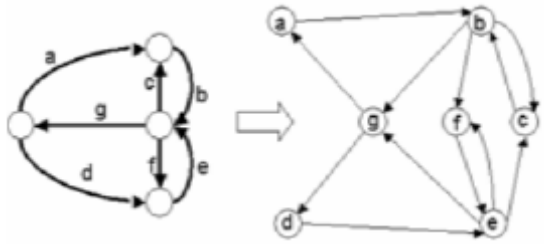

Fig. 2 Dual graph generation.

The graph is first converted into a binary string as shown in Fig 1. Next, the arcs of an original graph are converted into nodes as shown in Fig 2. As dual graph is traversed, all possible two links combination in dual graph for example bc, bf, bg ... . All the dual combinations are then encoded in 0 and 1 format as genetic population.[11][12]

APPLICATIONS OF GA IN BLACK BOX TESTING:

Black box testing is a useful testing. Here test cases are known from functional necessities of the tested system, which is viewed as a mathematical function mapping its input onto its output. While the number of possible box tests for any non-trivial program is extremely large, the testers can run only a limited number of test cases under their resource limitations. The effective set of test cases is one that has a high possibility of detecting faults presenting in a computer program.Francisca Emanuel has presented GA based technique to generate good test plans for functionality testing in an dispassionate manner to avoid the expert'sintervention.The test plan or test sequence totally relies on the experts or the people who understand the application well. The fact that "an error in a program is not necessarily due to the last operation executed

by the user but may have been due to a sequence of previously executed operations that leads an application in an inconsistent state".

As shown in Table 1, the transitions of an operation li yield a new operation li+1 which leads the system into a new state, the objective is to find the sequence of operations which leads the system in an inconsistent state. Fitness value for the Table 1 is calculated as:-

$$fp = \sum_{i}^{k-1} t(l_i \rightarrow l_{i+1})$$

	l_1	l_2	l_3	.....	l_n
l_1	$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	.....	$v_{1,n}$
l_2	$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	.....	$v_{2,n}$
l_3	$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	.....	$v_{3,n}$
:	:	:	:		
l_n	$v_{n,1}$	$v_{n,2}$	$v_{n,3}$	.....	$v_{n,n}$

Table 1: Representation of the assigned values for the inconsistency added by each transition for instance $t(l_2 \rightarrow l_3) = V2,3$.

Where p = l1, l2,...., lk is a test plan or sequence of operations and t is a transition function for converting one operation li to the next operation li+1 in a sequence or in a new state. The GA is applied on the table of size 30*30 with randomly generated transitions values as shown in Table 1. The results have shown that the GA improves the quality of the test plans.[13][14]

CONCLUSION:

Genetic algorithms are often used for optimization problems in which the evolution of a population is a search for a satisfactory solution given a set of constraints. We have reported preliminary results from an experiment comparing random test data generation with a new come within reach of using genetic search. In this paper, applications of GA in different types of software testing are discussed. The GA is also used with fuzzy as well as in the neural networks in different types of testing. It is found that by using GA, the results and the performance of testing can be improved. Our future work will involve applying GA selected paths in larger test data and further refining the method presented. This research would help in generating various software test cases. In future, we plan to use GA along with other soft computing techniques like fuzzy logic or neural networks for test case generation from UML diagrams. Also, since GA can be used independently for any problem and it is an emerging field so it has tremendous importance for users.

REFERENCES:

1. Somerville, I., "Soft ware engineering," 7th Ed. Addison-Wesley,
2. Deb, K., Agrawal, S.: Understanding interactions among genetic algorithm parameters. In: Banzhaf, W., Reeves, C. (eds.) Foundations of Genetic Algorithm, vol. 5, pp. 265–286. Morgan Kaufmann, San Francisco (1998)
3. D.J Berndt, A. Watkins, "High volume software testing using genetic algorithms", Proceedings of the 38t h International Conference on system sciences (9), IEEE, 2005, pp. 1- 9.
4. Francisca Emanuelle et. al., "Using Genetic algorithms for test plans for functionnal testing", 44th ACM SE proceeding, 2006, pp. 140 - 145.
5. Wegener, J., Baresel, A., and Sthamer, H, "Suitability of Evolutionary Algorithms for Evolutionary Testing," In Proceedings of the 26th Annual International Computer Software and Applications Conference, Oxford, England, August 26-29, 2002.
6. B. Korel. Automated software test data generation. IEEE Transactions on Software Engineering, 16(8), August 1990.
7. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice-Hall Inc., Englewood Cliffs (1995)
8. Last, M., Eyal, S.: A Fuzzy-Based Lifetime Extension of Genetic Algorithms. Fuzzy Sets and Systems 149(1), 131–147 (2005).

9. Liang You, YanSheng Lu, "A genetic algorithm for the time – aware regression testing reduction problem", International conference on natural computation, IEEE, 2012, pp. 596 – 599.
10. McMinn, "Search based software test generation: A survey", Software testing, Verification and reliability 14(2), 2004, pp. 105-156.
11. Timo Mantere, "Automatic software testing by Genetic Algorithms" Phd thesis, University of Vaasa, Finland, 2003.
12. Velur Rajappa et. al., "Efficient software test case generation Using genetic algorithm based graph theory" International conference on emerging trends in Engineering and Technology, IEEE, 2008, pp. 298 - 303.
13. Lin, J.C. and Yeh, P.L, "Using Genetic Algorithms for Test Case Generation in Path Testing," In Proceedings of the 9th Asian Test Symposium (ATS'00). Taipei, Taiwan, December 4 6, 2000.
14. Xuan Peng, Lu Lu, "A new approach for session - based test case generation by GA". IEEE, 2011, pp. 91- 96.
15. Christoph C. Michael, Gary E. McGraw, Michael A. Schatz, and Curtis C. Walton, "Genetic Algorithmsfor Dynamic Test Data Generation," Proceedings of the 1997 International Conference on Automated SoftwareEngineering (ASE'97) (formerly: KBSE) 0-8186-7961-1/97 © 1997 IEEE.
16. C. Darwin. On the Origin of Species: A facsimile of the first edition. Harvard University Press, July 1975.