Volume 4, Issue 12, December 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# A Review on Automated Test Case Generation Using Genetic Algorithms

[1]Rijwan Khan*, [2]Dr. Mohd Amjad
[1]Research Scholar, [2]Assistant Professor
Department of Computer Engineering, Jamia Milia Islamia- A Central University,
New Delhi, India

***Abstract:*** ***Software testing is a key of guaranteeing software quality and reliability. For testing software performance we have to need testing data or test case. So generation of test cases is one of the key steps for software testing. In generation of suitable test cases for a software/program, nature inspired algorithms play an important role. We studied several nature inspired algorithms but in this paper we will discuss only genetic algorithm. Our paper is a review paper for automatic test case generation for software testing using genetic algorithms.***

***Keywords:*** ***Automatic test case generation, Software testing, test cases, genetic algorithm.***

## I. INTRODUCTION

Software testing is an important phase of software development life cycle. Before delivering the software to the costumer there are some types of testing which has been done in the industries. These testing are unit testing, integrate testing, system testing. Here our focus in on unit testing. This unit testing is white box testing (WBT). So for the white box testing, researchers are using path coverage testing. Industries have spent a lot of its time and cost in testing of their software. Basically testing is depending on the test cases (input for the software). So to find the suitable test cases for software is a difficult task. Randomly generated test cases takes a lot of time to test the software. To reduce the time of the testing process automatic test case generation is a good option. In automatic test case generation, genetic algorithm plays an important role. With the use of the genetic algorithm we can generate suitable test cases.
Software testing also increases the functionality of the software. It also checks verification and validation of the software. Software testing is not only used for the removing the bugs and deleting the errors, it also insures to deliver error and fault free software to the customers. Software testing is static or dynamic. In static software testing the white box testing is used while dynamic testing is concern with the black box testing. In dynamic testing we give input and concern about only the outputs.

## II. GENETIC ALGORITHMS

In 1975, Holland published a book entitled "Adaptation in natural and artificial systems". In this book he presented the idea of genetic algorithm. Holland described that, how one can apply natural evolution algorithms for optimization problems? He constructed the first Genetic Algorithms in his work. The theory which Holland proposed has been further developed by researchers and now Genetic Algorithms (GAs) stand up as a powerful tool for solving search and optimization problems. Genetic algorithms principle is based on evolution and genetics. In his book, Holland proposed GA is a heuristic method that based on "Survival of the fittest". A genetic algorithm is a nature inspired evolutionary algorithm in which we solve optimization problem. Genetic algorithm is used to find approximate solutions to optimization problems. Genetic algorithm handles a population of possible solutions. In genetic algorithm each solution is represented through a chromosome. The genetic algorithm loops over an iteration process to make the population evolve. In the each iteration we have the following steps:
Step 1: Initialize population;
Step 2: Evaluate population;
Step 3: While Termination Criteria Not Satisfied repeat step 4 to 6
Step 4: Select parents for reproduction;
Step 5: Perform recombination and mutation;
Step 6: Evaluate population;

SELECTION: The first step is selection in which we select individuals for reproduction. In the selection process there are different methods to select the individual. Roulette wheel selection, linear ranking and tournament selection methods are applied for selecting the individual. With their fitness function we select the best individual for reproduction so that they will be strong individuals.
REPRODUCTION: In the second step, offspring are bred by the selected individuals. The algorithm can use both recombination and mutation for generating new chromosomes.

 

EVALUATION: In evaluation the fitness of the new chromosomes is evaluated.
REPLACEMENT: During the replacement, individuals from the old population are killed and replaced by the new ones [18, 19].

## III. GENETIC ALGORITHM IN SOFTWARE TESITNG

The key problem in software testing is to generate test case and its automation. It improves the efficiency and effectiveness and lowers the high cost of software testing [12].Simple random method is not enough to generate adequate amount of test data. Therefore, there is a need for generating test data using search based techniques [2]. In these search based techniques genetic algorithm is more efficient than climbing method and random method in generating test data [17].
In 2012, Hiroide Haga and Akihisa Suehiro [6] applied genetic algorithm and mutation analysis to generate automatic test cases. First they generated test cases randomly in first step and then refined these test cases using genetic algorithm. The process which they apply is as follows:

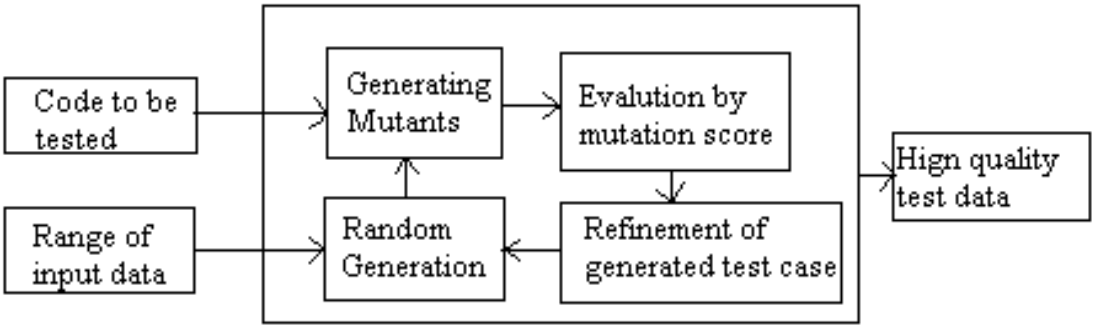


Figure 1 (Mutation based test case generation)

Random test cases are generated under the range of input data. Mutants are also generated from the code to be tested. Then the mutation scores are calculated by using the relation as mutation score (MS) = detected mutants/all mutants. If the mutation score is sufficient, the generated test case is the final result of the test cases. Otherwise, the generated test cases are refined using GAs. This iteration will be terminated when test case sets have been generated with sufficient mutation scores.
In 2006, Janvi Babdlaney *et. al.*[7] presented a paper on an introduction to data flow testing. In his paper, they generated the idea of a control flow testing. According to them control flow diagrams are a keystone in testing the structure of software programs. By examining the flow of control between the various components, they designed and selected test cases. Data-flow testing is a control-flow testing technique which also examines the life cycle of data variables. The main goal of their paper is to discuss the concept of data-flow testing and applying it to a running example.
In 2011, Jungsup Oh *et. al.* [8], used a messy-GA for transition coverage of Simulink / stateFlow models. They introduced a tool that implements their approach and evaluate it on three benchmarks embedded system Simulink models. Their messy-GA is able to achieve statistically better coverage when compared to both random search and to a commercial tool for Simulink/ State Flow model testing.
Sandra Rapps *et. al.*[15] defined how one can select suitable test data for a program using data flow information. They proposed a method similar to compiler optimization. In their method they used variable definition to use these variables. They include each point where variables are defined and variables are used.
In 2003, Nguyen Tran Sy *et. al.* [13] presented an approach for inter-procedural test data generation of imperative programs. These interactive programs contain integer variables, float and Boolean variables. Their main model works on maximum path coverage and first they find the appropriate path. There test program is represented by interprocedural control flow graph (ICFG). In their approach, they generate test cases by using the algorithm which they proposed for maximum path cover.
Michael *et. al.,* [11] proposed that the combinatorial optimization techniques when used with Genetic algorithms solve problems involving the simultaneous satisfaction of many constraints like, condition based decision coverage. The authors have performed experiments for comparing random test data generation with GA by applying genetic search. The genetic search outperformed random test generation, but the researcher experiment did not include the implementation of path selection heuristic for the generation of test data for programs with procedures.
H.Turgut *et al.* [3] proposed a method of sequence comparison for Evolutionary software test data generation. They proposed a method based on the pair wise sequence comparison techniques. To convert the generated path into the desired path they define a sequence distance which measures the amount of charge that needed for it.
Kobi Inkumsah (2008) *et.al.* [9] said that achieving high structural coverage is a challenging task. This high structural coverage such as branch coverage is more challenging task due to complexity of branch statement in object oriented programs. They made a tool called Evacon. The Evacon integrates symbolic execution and evolutionary testing. Their branch coverage compares is simply generated by evolutionary algorithms testing.
In 2009, Srivastava and Tai-hoon [16] find that it is most critical for improving software testing efficiency with the simple method so they applied Genetic Algorithm technique for the same. The authors present a method for optimizing software testing efficiency by identifying the most critical path clusters in a program. This is done by developing variable

 

length Genetic Algorithms that optimize and select the software path clusters, which are weighted in accordance with the criticality of the path. The approach used by the authors is a Weighted Control Flow Graph to test data generation using Genetic Algorithm. Path testing searches the program domain for suitable test cases that cover every possible path in the software under test. Path testing selects a subset of paths to execute and find test data to cover it. The authors conclude that Genetic Algorithm techniques can be applied to find the most critical path for improving software testing efficiency. The Genetic Algorithms also outperform the exhaustive search and local search techniques. The experiments conducted by the authors were based on small examples.

In 2009, Praveen Ranjan Srivastava *et. al.*[14], describes that generation of feasible test data genetic algorithm is most important algorithm. They proposed a system that was developed in java. They first generate initial population of test data and calculate their fitness function to generate next generation using crossover and some mutation method of genetic algorithm.

Hadi Hemmati *et. al*. [4] discussed a Model-based Testing (MBT). For model based testing they defined an enhanced test case selection approach. Model based testing (MBT) allows an automatic test case generation at very large level. In his work they proposed a technique whose function triggers and guards on transition of state machines and a genetic algorithm based selection algorithm.

In 2012, Binitha S.,*et. al.* [1] describes that, nature plays a great and immense role to solve the complex and hard problems of computer science. There are some problems that are NP-problems which simply cannot solved. Thus for solving these problem nature inspired algorithms are Meta heuristics which are used for finding optimal solutions of these problems. In his paper author discussed some nature inspired algorithms as Generic Algorithms (GA), Genetic Programming (GP), Different Evolution (DE), Evolution Strategies (ES) and Paddy Field Algorithm (PFA). They also discussed about swam intelligence algorithms as practical Swam Optimization (PSO), Artificial Bee colony Algorithm (ABC), Ant Colony Optimization (ACO), Intelligent water Drops Algorithm (IWD), Fish Swam Algorithm (FSA), Group Search Optimization (GSO), Firefly Algorithm (FA), Artificial Immune System Algorithms (AIS), Shuffled Frog Leaping Algorithm (SFLA) etc.

K. Kothewara Rao et. al. [10] presented basically one part of the program is tested by the researches which cannot give all the errors present in the code so author used GA for finding the most critical path cluster. Author assign weight to each path, the path that is most critical is assign high weight. For the long program they selected total weight as 100 and distributed among all the paths according to the criticality of path. For the small program they select total weight as 10 and out of 10 the total weight has been distributed to control flow graph of the program.

Harsh Bhasin *et. al*. [5] explained that test data generation is an exorbitant, error prone and boring task. Therefore, there is an immediate need to make the automation of process to generate the test cases efficient and powerful as possible. The work presented was intended for automation the process of test generation with a goal of attaining maximum coverage. The approach which they applied has been verified on programs selected in accordance with their lines of code and utility. The results which obtained in their experiment have been verified. Their proposed work can be used to develop a larger system and the task of that system is to check with white and black box testing respectively.

## IV. CONCLUSION

A number of optimization techniques have been applied for test case generation but no one could achieve the best performance for every piece of code. Since, test case generation has become an optimization problem hence scope remains open to apply some more techniques to achieve better results. In this paper, through discussion about GA as optimization techniques for test case generation is covered, which will pave the path for further work in this direction.

## REFERENCES

[1] Binitha S, S Siva Sathya, A Survey of Bio inspired Optimization Algorithms, international journal of soft computing and engineering(IJSCE) ISSN: 2231-2307, Volume-2, issue-2, May 2012.

[2] Ghiduk, Ahmed S and Girgis, Moheb R.(2010), Using Genetic Algorithms and Dominance Concepts for Generating Reduced Test Data, Informatica (Slovenia), Volume 34, Number 3, pp.377-385.

[3] H. TurgutUyar, A. SimaUyar, and EmreHarmanci (2007), Evolutionary Software Test Data Generation Using Sequence Comparison, international journal of principles and applications of information science and technology, December 2007, Vol 1, No1.

[4] HadiHemmati, Lionel Briand, AndeaArcuri, Shaukat Ali, An Enhanced Test Case selection Approach for Model-based Testing: An Industrial Case Studey, ACM, FSE-18, November 7-11, 2010, Santa Fe, New Mexico USA.

[5] Harsh Bhasin, NehaSingla, Shruti Sharma, Cellur Automata Based Test Data Generation, ACM SIGSOFT Software Engineering Notes, July 2013 Vol 38, No4.

[6] Hirohide Haga, Akihisa Suehiro, Automatic Test Case Generation based on Genetic Algorithm and Mutation Analysis, 2012 IEEE International Conference on Control System, Computing and Engineering, 23 - 25 Nov. 2012, Penang, Malaysia.

[7] Janvi Bandlaney, Rohit Ghatol, Romit Jadhwani, An Introduction to Data Flow testing, NCSU CSC TR-2006.

[8] Jungsup Oh, Mark Harman, Shin Yoo, Transition Testing for Simulink/Stateflow Model Using Messy Genetic Algorithms, ACM, GECCO'11, July 12-16, 2011, Dublin Ireland.

[9] Kobi Inkumsah, Tao Xie, Improving Structural Testing of Object- Oriented Programs via Integrating Evolutionary Testing and Symbolic Execution, IEEE 2008.

 

[10] Last Mark, Shay Eyal and Abraham K,(2005), Effective Black-Box Testing with Genetic Algorithms, Haifa Verification Conference, pp.134-148.

[11] M.Parthiban, M.R.Sumalatha, GASE -An Input Domain Reduction and Branch Coverage System Based on Genetic Algorithm and Symbolic Execution.

[12] Nirpal, Premal B. and Kale, K.V. (2010), Comparison of Software Test Data for Automatic Path Coverage Using Genetic Algorithm, Internal Journal of Computer Science and Engineering Technology, ISSN:2229-3345 Vol. 1, Issue 1.

[13] Peng Lin, Xiaolu Bao, Zhiyong Shu, Xiaojuan Wang, Jingmin Liu, Test Case Generation Based on Adaptive Genetic Algorithm, 978-1-4673-0788-8/12/ ©2012 IEEE

[14] Rajappa V. , Biradar A.and Panda S.(2008), Effective Software Test Case Generation Using Genetic Algorithm Based Graph Theory, First International Conference on Emerging Trends in Engineering & Technology, pp.298-303.

[15] Srivastava P.R and Tai-hoon Kim(2009), Application of Genetic Algorithm in Software Testing, International Journal of Software Engineering and its Applications, October 2009, Vol. 3, No. 4.

[16] Yang Cao, Chunhua Hu and Luming Li(2009), An Approach to Generate Software Test Data for a Specific Path automatically with Genetic Algorithm, International Conference on Reliability, Maintainability and Safety, pp.888-892.

[17] K. KoteswaraRoa, G. S. V. P.Raju, SromovasanNagraj, Optimizing the Software Testing Efficiency by Using a Genetic Algorithm- A Design Methodology, ACM SIGSOFT Software Engineering Notes May 2013 Volume 38 Number 3.

[18] S. N. Sivanandam, S. N. Deepa, Introduction to Genetic Algorithm, Springer 2008, book.

[19] Rijwan Khan, Mohd Amjad, Automated Test Case Generation using Nature Inspired Meta Heuristics- Genetic Algorithm: A Review Paper, International Journal of Application or Innovation in Engineering & Management (IJAIEM) Volume 3, Issue 11, November 2014 ISSN 2319 – 4847.

[20] John H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, 1975.

 