A Survey on Generation of Test Cases and Test Data Using Artificial Intelligence Techniques

Shveta Parnami, Prof. K.S.Sharma
Department of CS and IT
The IIS University
Jaipur, Rajasthan, India
shvetaparnami@gmail.com

Dr. Swati V.Chande,
Department of Computer Science
International School of Informatics and Management
Jaipur, Rajasthan, India

***Abstract*-Testing plays an important role in software development life cycle. Software testing is a critical element in software quality assurance and represents the ultimate review of specifications, design and coding. It is in general a laborious, costly and time consuming task: it spends almost 50% of software system development resources. For a good test quality the systematic design and appropriate selection of test cases and test data is essential. Test cases and test data generation is a key problem in software testing and its automation improves the efficiency and effectiveness and reduces the high cost of software testing. The application of Artificial Intelligence techniques in Software Testing is an emerging area of research that brings about the cross fertilization of ideas across two domains. Artificial Intelligence techniques of searching are used to automate test data and test cases. The paper presents an analysis of relative efficiency in using test data and test cases using artificial intelligence techniques.**

***Keywords*-Test data, test cases, artificial intelligence techniques, genetic algorithm.**

I. INTRODUCTION

A. Software Testing:

Software testing is the process of exercising and evaluating a system or system component by either manual or automated means to verify that it satisfies specified requirements and identifies the differences between the expected and actual results. It is performed for defect detection and reliability estimation. The software testing is conducted by executing the program developed with test inputs and comparing the observed output with the expected one. The main aim of testing is to cover the programming features. In White-box or structural testing, test data is design for program coverage. That means all paths of program should be executed at least once. There are three main types of coverage criteria; statement coverage, branch coverage, and path coverage. Branch coverage is widely used testing technique and it is the basis of several industry standards because it is not an extremely strict coverage criterion [14]. The Black-box or functional testing does not need any information about how the program was written. It generates test from software specification to ensure that software work properly. Gray-box testing investigates the coverage criteria of white-box method and finds all possible coverage paths.

It is difficult to test the whole software, therefore the selective parts of the software are considered for the testing. Because the input space of the Software Under Test (SUT) might be very large, testing has to be conducted with a representative subset of test cases. The test cases defined decide about the kind and scope of the test. Creation of relevant subset of test cases during software testing is a critical activity [13]. The test cases which are used to examine the SUT must possess an ability to expose the faults as well as test cases must be a representative subset of possible inputs. The quality and the significance of the overall test are directly affected by the set of test cases that are used during testing. Test data is used to create the test cases. Test data generation is the process of identifying a set of program input data that satisfies a given testing criterion [1]. This requirement of effective test cases demands the generation of 'Good' automated test cases. Test da generation technique and application of a test data adequacy criterion justifies the 'Better' test data[1]. There is need to explore these aspects of test data generation in order to increase the degree of automation and efficiency of software testing.

B. Artificial Intelligence:

Artificial Intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs. Artificial Intelligence (AI) is defined as the ability of computer software and hardware to do those things that we, as humans, recognize as intelligent behavior. These include activities as:

- Searching: finding "good" material after having been provided only limited direction, especially from a large quantity of available data.
- Surmounting constraints: finding ways that something will fit into a confined space, taking apart or building a complex object, or moving through a difficult maze.
- Recognizing patterns: finding items with similar characteristics, or identifying an entity when not all its characteristics are stated or available.
- Making logical inferences: drawing conclusions based upon understood reasoning methods such as deduction and induction.

In AI, these processes have manifested themselves in well-recognized and maturing areas including Neural Networks, Expert Systems, Automatic Speech Recognition, Genetic Algorithms, Intelligent Agents, Natural Language Processing, Robotics, Logic Programming, and Fuzzy Logic.

II. REVIEW ON APPLICATION OF ARTIFICIAL INTELLIGENCE TECHNIQUES IN SOFTWARE TESTING

One of the software engineering areas with a more prolific use of artificial intelligence techniques is software testing. According to [14] the test data are generated either with static method or dynamic method. Static methods include the domain reduction and symbolic execution and the dynamic methods include random testing, local search approach, goal oriented approach, chaining approach and evolutionary approach. Static methods suffer from a number of problems when they handle indefinite array, loops, pointer references and procedure calls whereas the dynamic test data generation avoids these problems. AI techniques used for test data generation included the AI Planner Approach, Simulated Annealing, Tabu Searching, Genetic Algorithm and Ant Colony Optimization (ACO). Being a robust search method in complex spaces, genetic algorithm is applied to test data generation and evolutionary approach has become a burgeoning interest since then.

Premal and Kale in [13] applied the AI technique to find the most critical path clusters in a program for improving software testing efficiency. The weighted control flow graphs are created for test data generation using Genetic Algorithm. Huaizheng and Lam in [7] have used an Ant Colony Optimization approach to automatically generate test data from UML Statechart diagrams for state-based software testing. The proposed approach deals with the automatic generation of test suites from the UML Statechart diagrams for state based software testing, and uses the all state testing coverage as test adequacy requirement.

In recent years, advanced heuristic search techniques have been applied to software testing. These techniques are based on evolutionary algorithms. Software testing uses a meta-heuristic search technique, to convert the task of test case generation into an optimal problem [6]. Faezeh et. al. in [5] focused on the use of independent path to reduce time and on precisely monitoring the execution trace of the program. Genetic algorithm is applied with improved parameters for test cases designed to better detect bugs of tested program. The genetic algorithm based tester fulfills test criteria by manner of evolutionary computation. Genetic Algorithm method with dynamic fitness function and stopping criterion is used for effective testing and low cost identification of infeasible path. The approach used suffers from the disadvantage about dynamic aspect of testing, as the stopping criteria used can't specify actual number of generations, i.e. in some cases, the tester is exited based on waiting time, while the stopping criterion is not satisfied.

Generating test data automatically and identifying infeasible paths reduces the testing cost, time and effort. Anu et. al. in [2] mentioned that the work has been done on the automation of test cases using Tabu search algorithm on complex programs under test and large number of input variables. The tabu search algorithm is used for generation of structural software tests. The authors present use of tabu search with dijksra algorithm (a greedy approach) to provide an efficient path with maximum code coverage and minimum cost.

III. DISCUSSION ON RESULTS

Software testing takes a large portion of the software project resources. Reduction in cost and time at this stage will be of great help for software development process. Test cases and test data generation is a key problem in software testing and its automation improves the efficiency and effectiveness and improves the high cost of software testing [14].

In most test case design it is difficult to automate e.g. for functional testing the generation of test cases is usually not possible as no formal specifications are applied in industrial practice and for structural testing the limits of symbolic execution make an entire automation impossible. Many researchers and practitioners have been working in generating optimal test cases based on specifications [8]. Parsana and Chandran in [11] had suggested a model based approach to derive test cases using tree structure coupled with genetic algorithm and have concluded that their proposed model is useful to generate test cases. The genetic algorithm is also used to automatically generate test cases for path testing [13].

The generations of test data using random, symbolic and dynamic approach are not sufficient enough to generate adequate amount of test data. Other problems like non-reorganization of occurrences of infinite loops and inefficiency to generate test data for complex programs makes these techniques not worthy for generating test data. Therefore there is need for generating test data using search based technique [1]. In addition to these there is need of generating test cases that concentrate on error prone areas of code [3].

Test data is often generated by hand, so demand for automatic test data generation is high in these sectors. Ahmed and Moheb in [1] showed through experiments that the test data generation based on search techniques like genetic algorithm reduced the cost of software testing by more than 75% and when random test data generation is compared with an approach based on genetic search then it has been found that genetic search visibly outperformed random test generation [14]. Premal and Nirpal in [13] concluded that when Genetic Algorithm techniques applied for finding the most critical paths in order to generate the test data outperforms the exhaustive search and local search techniques. Using the Ant Colony Optimizer algorithm, a group of ants effectively explore the graph created to represent the Statechart model structure of a software system under test and generate optimal test data to achieve test coverage requirement [7]. Random test generated test data do not give a good test data set. The quality of test data produces by genetic algorithms is higher than the quality of test data produced by random way because the algorithm can direct the generation of test cases to the desirable range fast [4]. Genetic algorithms are also useful in reducing the time required for lengthy testing by generating meaningful test data. [13]

IV. CONCLUSION

Software testing is the process of identifying the flaws and defects in the system. The goal of software testing is to design a set of minimal number of test cases and test data such that it reveals as many faults as possible. In software testing process each test case has an identity and is associated with a set of inputs and a list of expected outputs. Test data generation is the process of identifying a set of program input data that satisfies a given testing criterion. Test cases and test data generation is a key problem in software testing and its automation improves the efficiency and effectiveness and improves the high cost of software testing. The automation of test data and test cases generations using artificial intelligence techniques like genetic algorithm, simulated annealing and ant colony optimizer are better than the generation of test data and test cases using the exhaustive and random test generation.

V. REFERENCES

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

2. Anu Sharma, Arpita Jadhav, Praveen Ranjan Srivastava, Renu Goyal (2010), Test Cost optimization Using Tabu Search, Journal on Software Engineering and Applications, 2010, Vol. 3, pp. 477-486.

3. Berndt, D.; Fisher, J.; Johnson, L.; Pinglikar, J. and Watkins, A.(2003), Breeding Software Test Cases With Genetic Algorithms, Proceeding of the 36th Annual Hawaii International Conference on System Sciences, Track 9, Volume 9, ISBN: 0-7695-1874-5, pp.10.

4. C Doungsa-ard, K Dahal, and A Hossain ( ), AI Based Framework for Automatic Test Data Generation, International Conference on Software Engineering Advances, 2007. ICSEA 2007., 25-31 Aug. 2007, ISBN: 0-7695-2937-2, pp. 47.

5. Faezeh S. Babamir, Esmaeil Amini, S. Mehrdad Babamir, Ali Norouzi and Berk Burak Ustundag(2010), Genetic Algorithm and Software Testing based on Independent Path Concept, International Conference on Genetic and Evolutionary Methods-GEM'10, The 2010 World Congress in Computer Science, Computer Engineering and Applied Science, Las Vegas, Nevada, USA, July 2010.

6. Gupta, N.K. and Rohil M.K.,(2008), Using Genetic Algorithm for Unit Testing of Object Oriented Software, International Conference on Emerging Trends in Engineering and Technology, 16th -18th July 2008, Nagpur, Maharastra, ISBN: 978-0-7695-3267-7, pp. 308-313.

7. Huaizheng Li and C. Peng Lam (2005), Software Test Data Generation using Ant Colony Optimization, Proceedings of World Academy of Science, Engineering and Technology, Vol. 1, Jan 2009. ISSN 1307-6884.

8. Joachim Wegener (2001), Evolutionary Testing, Testworkshop TUM, 18th Jan 2001, Germany.

9. Michael, C.C., McGraw, G.E., Schatz, M.A. and Walton, C.C. (1997), Genetic algorithms for dynamic test data generation, International Conference on Automated Software Engineering, 1st – 5th November 1997, Incline Village, NV, USA, ISBN: 0-8186-7961-1, pp. 307-308.

10. Misra A, Mehra R, Singh Mayank, Kumar Jugnesh and Mishra Shaailendra(2011), Novel Approach to automated test data generation for AOP, International Journal of Information and Education Technology, June 2011, Vol.1.

11. Parsama M and Chadran K.R (2011), Automatic Test Case Generation for UML Object Diagrms using Genetic Algorithm, Internal Journal of Advance Soft Computing Application, July 2011, ISSN-2074-8523, Vol.1,

12. Premal B. Nirpal 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. Premal B. Nirpal and Kale K.V.(2011), Using Genetic algorithm for automated software test case generation for path testing, Internal Journal of Advanced Networking and Applications, Vol. 2, Issue 6, pp.911-915.

14. 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.