Volume 4, Issue 6, June 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 Optimization Methodologies Used for Randomized Unit Testing

K. Devika Rani Dhivya,*
Msc, Mphil
Asst.Professor, Department of CA & SS,
Sri Krishna Arts and Science College,
India

C. Sunitha,
MCA., M.Phil
Head, Department of CA & SS,
Sri Krishna College of Arts and Science,
India

Abstract- This paper gives a review on various methods to find the best optimal solution in minimum time period for Randomized unit testing. Randomized unit testing (RUT) is an effective method for testing a software unit. This review carried out by analyzing many papers which is related to Genetic Algorithm (GA) for Randomized unit testing. GA is an optimal search algorithm, which is used to find best optimal parameter for large methods. A number of studies have been done to improve the time efficiency, speed of the system and optimization process by using randomized testing with GA.

Keywords- Randomized unit testing, Genetic algorithm.

I. INTRODUCTION

Software testing is the process of checking the correctness of the software. It is a process of executing a program with the intention of finding errors. Testing is one of the vital activities which must perform during the software development. It is conducted by executing the program developed with the test input with the expected value. Randomized unit testing is a way testing a unit. It uses randomization for some aspects of the test input data selection. RUT uses coverage models that contain a bias on the random variables which select for the target method call sequence and/or arguments to the method calls. A number of studies have been shown less time efficiency, less optimization process and Test case generation by using randomized unit testing with genetic algorithm. The problems mainly focused in this paper are due to the iteration process it becomes too complex and time consuming. It does not cover of all test cases in the system and values of the parameters are not optimal. It slows down the processing of the system. It does not minimize the test case generation and it will not quickly generate the test case.

II. BACKGROUND STUDY

A new system of utilizing program dependence analysis techniques and GA is used to generating test data. Here they use a set of the new approaches where used to show its effectiveness and efficiency based on some criteria. That improves the effectiveness of test case generation with optimal parameter. [1]

The randomized testing, which is an effective method for testing the unit of software. They describe a system which uses a "Genetic algorithm " to find out the parameter for the randomized unit testing that optimized the test coverage. Here, they describe about the implementation of "A novel two-level genetic random testing system "Nighthawk". They compare Nighthawk with purely genetic algorithm approach which is the previous work, and it shows that 100% of feasible condition and decision coverage. They proposed "A two-level genetic – randomized unit testing system", Nighthawk. Here, they concluded that randomized unit testing is a promising technology that has been too effective, but whose thoroughness depends up on the settings of the testing algorithm parameters, thus they describe the nighthawk, a system which an upper level genetic algorithm and lower level randomized unit testing concept. This tool, which achieves a good test coverage and produce optimal result. [2]

The tools developed for automating the process of unit testing in Sulu, the main features of the Sulu language. An experiment describes to assess the practical effectiveness of their approach, by running automatically generated test suites against a reference set of small software components. The main advantage of this technique is automating the generation, execution, and evaluation of test suites using both code coverage and mutation analysis. In this method high mutation coverage for the most comprehensive test suite generated. [14]

A method of generate test case, a new approach to "automate the generation of object oriented unit test case". They proposed a method of generate test case for class in object oriented software using a genetic programming approach. An approach, which facilities the automatic generation of object-oriented test program using Genetic algorithm which was proposed to achieve automated generation of test case, they use an effective way to use Randomized test generation. [9]

A model generation is created to meet the size and time constraints and try to improve the speed and coverage of the system. There will always be a trade-off between completeness and runtime speed, thus they explore that trade-off in the context of using genetic algorithms. After applying feature subset selection to the GA output, they found that can generate the coverage model and run the resulting test suite 10 times faster while only losing 6% of the test case coverage. They applied FSS to a metaheuristic algorithm used to learn coverage models for randomized unit test. It cut downs on the number of gene types without a noticeable loss of coverage. Since each gene type corresponded to runtime cost, this also allowed us to cut down on the time and memory taken for the algorithm. Thus it improves the time efficiency. [3]

Genetic Algorithm is used to path-oriented test data generation and it outperforms other approaches. A fitness function based on branch distance (BDBFF) and another based on normalized extended Hamming distance (SIMILARITY) are both applied in GA-based path-oriented test data generation. Thus the results shows that BDBFF-based approach can generate path-oriented test data more effectively and efficiently than SIMILARITY- based approach does. [16]

A system which uses GA to find out the optimum values for randomized unit testing that optimizes test coverage. Designing GA with FSS tool is used to reduce the size and content of the representation within the GA. This tool "prunes" back 90% of GA's matadors while still achieving most of the coverage found. This pruned GA achieves almost the same result as the full system, but in only 10% of the time. Thus the result suggested that FSS is an effective pruning tool to get optimized search. They gave a clear view of Nighthawk, a system which consists of two levels, the upper level genetic algorithm which derives good parameters values automatically and lower level randomized unit testing algorithm. They conclude that Randomized unit testing is a promising technology that has been shown very much effective .This proposed technique can able to optimize and simplify meta-heuristic search tool. Here, the meta-heuristic search tool such as GA which typically mutate some aspect of the solution and evaluate the result. If the effect of mutating each aspect is recorded, then each aspect can be considered a Feature and it enables to the FSS processing. By this way FSS can be used to automatically find and remove the part of the search control. Technique of pruning the nighthawk improves the fast, optimization and the reliability of the program. [4]

A method developed for optimizing software testing efficiency by identifying the most critical path clusters in a program. They initialized this by developing variable length GA that optimize and select the software path clusters. Which are weighted i accordance with the criticality of the path. They are developed a more selective approach for testing by focusing on those parts that are most critical so that these paths can be tested first. By identifying the most critical paths, the testing efficiency can be increased. [10]

Investigating the performance of the proposed GA with different parameters combination used to automate the test data generator, the investigation involves crossover and selection of parent for reproduction and mutation. The result showed that double crossover is much better in path coverage. This study results also that, selecting parent for reproduction according to their fitness is more efficient than random selection. And that mutation rate is better adjusted with program at hand. Also, they studied the generation to generation progress in the proposed GA while searching for good test data. Their work is compared with random testing. And result that the proposed GA improves the search from one generation to the next, and performs better than random testing, where the search was absolutely random and does not show improvement through the generations. Another observation is that random testing generates less. [7]

Unit testing with static and dynamic analysis, is used by the programmer to discover the bugs with least cost. During the static check, besides finding program grammar errors, they can also detect the extra code, calculate the path to cover all branches in program and generate input test. In dynamic analysis all control flow and data flow will be exercised, and all boundary conditions and error handling path will be checked. In static analysis, this method can help testers to detect grammar errors, extra code, non-initialized variables, and search paths to cover the code, and generate tests. In dynamic, testers can get benefit from this method to check internal logic, exception and boundary conditions. [8]

An approach, automated test generation for EFSM models. Design by contract approach is applied to formalize specification requirements. Extended Finite State Machines (EFSMs) are often used in model-based development and for modeling VHDL specifications. Genetic algorithm is proposed to find set of values that triggers given path in the EFSM and reveals inconsistencies with the specification. This improves the efficiency of the model. [17]

A hybrid algorithm (GA-PSO) which combines Genetic Algorithm and Particle Swarm Optimization (PSO) the new algorithm is proved effective by a representative test of the "triangle type of discrimination". The experiment shows that the new algorithm has higher performance value of 20%. [13]

Nighthawk, a system which uses a GA to find parameters for randomized unit testing that optimizes test coverage. GA uses a FSS tool to assess the size and content of test case. This tool can reduce the size of the test case and still achieving most of the coverage found using the full representation. Thus the reduced GA achieves almost the same results as the full system, but in only 10% of the time. These results suggest that FSS could significantly optimize meta-heuristic search-based software engineering tools. The Nighthawk unit test data generator. Randomized unit testing is unit testing where there is some randomization in the selection of the target method call sequence and/or arguments to the method calls. They have shown that Nighthawk is able to achieve high coverage of complex, real-world Java units, while retaining the most desirable feature of randomized testing: the ability to generate many new high-coverage test cases quickly. This technique achieves a good test coverage and produce optimal result. The optimization of the parameter is better while comparing with the previous

works. The drawback in this technique is that the coverage criteria are not is not good. The time taken to execute the parameters is also a drawback. [5]

An approach named US-RDG in terms of gray-box testing, combining User Session data with Request Dependence Graph (RDG) of web application, to automatically generate test cases with the using of GA. Simulation results indicate that US-RDG effects better than the traditional user-session-based testing, and attains higher path coverage and fault detection rate within small size of test suite. [15]

A technique for optimizing static testing efficiency by identifying the critical path clusters using genetic algorithm. The testing efficiency is optimized by applying the genetic algorithm on the test data. The test case scenarios are derived from the source code. The information flow metric is adopted in this work for calculating the information flow complexity associated with each node of the control flow graph which is generated from the source code. [11]

A technique that based on a combination of genetic algorithm (GA) and particle swarm optimization (PSO), and is thus called GPSCA (Genetic-Particle Swarm Combined Algorithm) which is used to generate automatic test data. The performance of the proposed approach is analyzed on a number of programs having different size and complexity. Finally, the performance of GPSCA is compared to both GA and PSO for generation of automatic test cases to demonstrate its superiority. [12]

A system which integrates the Feature Sub Set selection tool with genetic algorithmic. This type of integration is used to optimize the GA for randomized unit testing**.** This paper described a system in which an upper level GA automatically generates good parameter values for a lower-level RUT. They have shown that Nighthawk is able to achieve high coverage of complex, real-world Java units, while retaining the most desirable feature of randomized testing and the ability to generate many new high-coverage test cases quickly. FSS can be used to automatically find and remove superfluous parts of the search control this decreases the time of execution and it also provides a better optimal parameter. [6]

III. PERFORMANCE ANALYSIS

From the above review we analysis the tools with GA for Randomized unit testing had performed for improving the efficiency of time and to get best outcome for RUT and following the analysis performances periodically. The parameters which we are taken for this performance analysis are improved time efficiency and improved optimization. Some optimization technique and tools are used with genetic algorithm for improving the performance. Many tools where given a best result with better performance measure.

To get optimized result and to Improve time efficiency GA combined with different tools, they are the optimization techniques for GA methods tools like Nighthawk, Pure genetic algorithms (GAs) to generate test data, A novel two-level genetic random testing system Nighthawk, Pruned GA with FSS tool, FSS Learner into the genetic algorithmic level of Nighthawk, Improved GA with Nighthawk for randomized unit testing has been tableted in Table 1 for the performance analysis.

TABLE 1
PERFORMANCE ANALYSIS

Year	Title	Author	Methodology	Improved Time Efficiency	Improved optimization
2006	Automatic Test data generation using genetic algorithm and program dependence graph	James Miller, Marek Reformat, Howard Zhang	Pure genetic algorithms (GAs) to generate test data	10%	-
2007	A Two-Level Genetic-Random Unit Test Data Generator	James H. Andrews and Felix C. H. Li	A novel two-level genetic random testing system "Nighthawk.	10%	Yes
2008	Using Genetic Algorithm for Unit Testing Of Object Oriented Software	Nirmal Kumar G and Mukesh Kumar R	The automatic generation of object-oriented test program using Genetic algorithm	-	Yes

2009	Controlling Randomized Unit Testing With Genetic Algorithms	James H. Andrews, Tim Menzies, and Felix C. H. Li	Pruned GA with FSS tool	20%	Yes
2009	Application of Genetic Algorithm in Software Testing	Praveen Ranjan Srivastava1 and Tai-hoon Kim	Genetic algorithm	-	Yes
2010	Automatic path test data generation based on GA-PSO	Shenga Z, Ying Zhang, H Z & Qingguan H	Hybrid algorithm (GA-PSO)	20%	Yes
2011	Genetic Algorithms for Randomized Unit Testing	James H. Andrews, Tim Menzies and Felix C. H. Li	Improved GA with Nighthwak for randomized unit testing	10%	Yes
2012	Dynamic Optimization of Genetic Algorithms for Randomized Unit testing by using FSS	Jahnavi K and Basha P	FSS Learner into the genetic algorithmic level of Nighthawk	10%	Yes

IV. CONCLUSION

This paper mainly focuses on the randomized unit testing with genetic algorithm. This survey concludes that randomized unit testing with genetic algorithm is less time efficient, slows down the speed of processing and produces less optimized parameters. Randomized unit testing is a promising technology that has been shown to be effective, but whose thoroughness depends on the settings of test algorithm parameters. A number of studies have been shown less time efficiency, less optimization process and Test case generation by using randomized unit testing with genetic algorithm. The optimization techniques for GA methods tools like, A novel two-level genetic random testing system Nighthawk, Pruned GA with FSS tool, FSS Learner into the genetic algorithmic level of Nighthawk, Improved GA with Nighthawk for RUT are used but the optimization of testing process is not optimal and also produces less test case generation, system does not consider the fitness value for randomized unit testing it generates the random selection of software units. Due to the iteration process comes too complex and time consuming. It does not cover of all test cases in the system and values of the parameters are not optimal. It does not minimize the test case generation and it will not quickly generate the test case. To overcome these problems, we are going to assign PSO, weighted PSO, Bat inspire algorithm. Thus in future we can assign other best advanced optimization algorithm which is more efficient than the GA algorithm. This improves fast, time efficiency and optimization.

REFERENCES

1. James M, Marek, Howard Z, *Automatic Test data generation using genetic algorithm and program dependence graph*, Information and Software Technology Volume 48, Issue 7, July 2006, Pages 586–605.
2. James H. Andrews and Felix C. H. Li, *Nighthawk: A Two-Level Genetic-Random Unit Test Data Generator*, ASE'07, November 4–9, 2007, Atlanta, Georgia, USA. ACM 978-1-59593-882-4/07/0011
3. James H. Andrews, Tim Menzies, and Felix C. H. Li, *Controlling Randomized Unit Testing With Genetic Algorithms*,September 27, 2009.
4. James H. Andrews and Tim Menzies, *On the Value of Combining Feature Subset Selection with Genetic Algorithms: Faster Learning of Coverage Models*, University of Western Ontario, Published by ACM 2009 Article.
5. James H. Andrews, Tim Menzies and Felix C. H. Li , *Genetic Algorithms for Randomized Unit Testing*, *IEEE transactions on software engineering*, vol. 1, no. 1, January 2011.
6. Jahnavi K and Basha P, *Dynamic Optimization of Genetic Algorithms for Randomized Unit testing by using FSS*, International Conference on Computing and Control Engineering (ICCCE 2012), 12 & 13 April, 2012.
7. Kumar A, Alzabidi M and A.D. Shaligram ,*Automatic Software Structural Testing by Using Evolutionary Algorithms for Test Data Generations*, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.4, April 2009

8. Na Z, Xiaoan B and Zuohua D, *Unit Testing: Static Analysis and Dynamic Analysis*, Computer Sciences and Convergence Information Technology, 2009. ICCIT '09. Fourth International Conference on 24-26 Nov. 2009 Page(s): 232 – 237.
9. Nirmal Kumar G and Mukesh Kumar R, *Using Genetic Algorithm for Unit Testing Of Object Oriented Software*, First International Conference on Emerging Trends in Engineering and Technology 978-0-7695-3267-7/08 $25.00 © 2008 IEEE DOI 10.1109/ICETET.2008.137
10. Praveen Ranjan Srivastava1 and Tai-hoon Kim, *Application of Genetic Algorithm in Software Testing*, International Journal of Software Engineering and Its Applications Vol. 3, No.4, October 2009.
11. Sabharwal, S. Sibal, R. ; Sharma, C. , *A genetic algorithm based approach for prioritization of test case scenarios in static testing*, 15-17 Sept. 2011 Page(s): 304 – 309
12. Sanjay Singla, Dharminder Kumar, H M Rai and Priti Singla, *A Hybrid PSO Approach to Automate Test Data Generation for Data Flow Coverage with Dominance Concepts*, International Journal of Advanced Science and Technology, Vol. 37, December, 2011.
13. Shenga Z, Ying Zhang, H Z and Qingguan H, *Automatic path test data generation based on GA-PSO*, Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on 29-31 Oct. 2010 Volume: 1 Page(s): 142 – 146.
14. Tan, R. P. and Edwards, S, *Evaluating Automated Unit Testing in Sulu*, Software Testing Verification an Validation 2008 is International Conference on
15. Date of Conference: 9-11 April 2008 Page(s): 62 - 71.
16. Xuan P and Lu L , *A new approach for session-based test case generation by GA*, Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on Date of Conference: 27-29 May 2011 Page(s): 91 - 96
17. Yong C , Yong Z ; Tingting S ; Jingyong L ,*Comparison of Two Fitness Functions for GA-Based Path-Oriented Test Data Generation*, Natural Computation, 2009. ICNC '09. Fifth International Conference on 14-16 Aug. 2009 Volume**:** 4 Page**(s):** 177 – 181
18. Zakonoy A; Stepanov, O ; Shalyto A , *GA-based and design by contract approach to test generation for EFSMs*, Design & Test Symposium (EWDTS), 2010 East-West Date of Conference: 17-20 Sept. 2010 Page(s): 152 – 155

