www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 5 May, 2014 Page No. 5642-5646

Enhance Performance of Random Testing using Randomized Algorithm

Jaypal D. Rangari[1]***, Swapnili P. Karmore***[2]

[1]Student, Department of Computer Science and Engineering,
G. H. Raisoni College of Engineering, Nagpur, India
jay.drs@gmail.com

[2]Professor, Department of Computer Science and Engineering,
G. H. Raisoni College of Engineering, Nagpur, India
swapravin@gmail.com

Abstract: ***this paper implements and enhances performance of software random testing. Random testing is a base software testing technique that can be used to improve the software reliability as well as to discover software failures. Random testing is a black-box software testing technique where programs are tested by generating random, independent inputs. In proposed methods uses both Monte Carlo and Las Vegas Randomized algorithms. Monte Carlo has fast execution while Las Vegas has low execution time, but sometimes Monte Carlo algorithm gives false result while Las Vegas gives always correct result. In proposed method has two result sets, in first result set has executed test cases and in second has fails test cases. Initially test cases are tested using Monte Carlo algorithm and produced executed and fail result sets. The fail result set is again tested using Las Vegas algorithm because sometimes Monte Carlo gives false result. We present a technique that improves performance random testing. These results are very hopeful, given that evidences that our perception is likely to be useful in improving the efficiency of random testing.***

Keywords: Testing, random testing randomized algorithm, performance of random testing, software testing

1. Introduction

Randomized algorithms are used to explain many software engineering problems. It uses degree of randomness as part of their logic. In random testing randomized algorithms are very significant for difficult problems where an exact result cannot be calculated in a deterministic way. However, randomized algorithms produce dissimilar results on every run when it applied to the same problem. It is very important to evaluate the effectiveness of randomized algorithms by collecting data from a large number of executions. It performs a complete study as well as gets complete information of current practice in software engineering research. The test case selection is simple and easy, they are randomly chosen among huge range. Random testing is more commercial in software testing problem. Effectively combining random testing with other testing techniques may provide up more powerful and cost-effective testing methods.

1.1 Specification-Based Techniques

- Equivalence partitioning
- Boundary value analysis
- Decision table
- Finite-state machine method
- Testing from formal specification
- Random testing

Software testing is an execution of the software or given code with the aim of removing and debugging failures which is an important phase to confirm the correctness of software system. In software testing consist of two important steps, i.e. 1) generating test cases and 2) validating the performance of software system by executing the test cases [1]. In general, since we cannot execute all the test cases, it is performed with limited test cases. Thus, the good quality of test cases results to the good quality of software products. Random testing is one of the most standard techniques in software testing although the random testing (RT) is easy for execution.

Testing a system is the procedure of finding errors which correct any gaps, errors or missing requirements. In the randomized algorithms apply a random value to solve various problems. Randomized algorithms are useful in difficult problems in an exact solution can be derived in a deterministic manner. Different outcome in each run occur when applied to the same problem instance. There are two types of randomized algorithm

1. Monte Carlo algorithm
2. Las Vegas algorithm

1.2 Monte Carlo algorithm:

It may produce incorrect results, but with very less error probability. It has a fixed execution time. If the algorithm is executed repeatedly with independent random options each time, the probability of failure can be made arbitrarily little at the cost of running time. In this algorithm with one-sided errors, the failure probability can be minimized by executing the algorithm k times [18]. Thus, prime numbers answer is always correct, and for composite number the answer is correct with probability at least 1−(1−1/2)k = 1−2−k.

1.3 Las Vegas Algorithm:

This algorithm always gives the correct result and its execution time is higher than Monte Carlo algorithm. For high complexity execute Las Vegas randomized algorithm because we need correct result after the execution.

Random testing is black-box software testing, it is not focus the code or methods that written in the program. In Fig.1 black box random testing, it focuses on inputs and output only. During execution of software testing randomized algorithm is used to generate test cases. It generates random number of choices during test cases execution to produce a result. It selects test cases from the whole input set randomly. The process of test cases generation can be minimized using randomized algorithm..

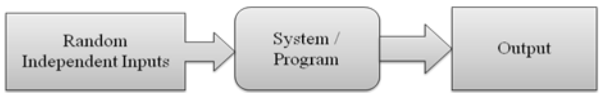


Figure. 1. Black box random testing.

1.4 Why we use random testing?

Random testing gives an advantage of easily calculating software reliability from test outcomes. Test inputs are randomly generated according to a prepared profile, and failure times are recorded. The data obtained from random testing can then be used to calculate reliability. Other testing methods cannot be used in this way to calculate software reliability. Use of random test inputs may save some of the time and effort. It must consider the time needed to write random test generator verses the time to write a set of directed tests. Random Testing methods are applicable for any single project. Different testing techniques can find different types of defects.

2. Implementation

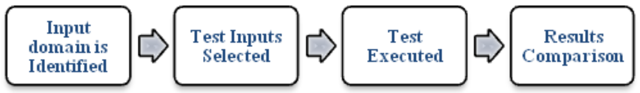


Figure. 2. Random testing implementations steps.

Random testing is also known as black-box software testing technique. In the figure 2 shows the complete execution of random testing.

2.1 Input domain identification

This is basic step in software random testing. There are various classes, functions and methods in the system or programs. First analyze the whole system and find out classes, function and methods separately, each one have different input domains. We need to identify the input domain according to the software requirement specification for each classes, methods and functions for testing. If we need to print even number from 1 to 10 then input domain will be 0 to 11 (InputD[0-11]).

2.2 Test input selection

After identification of input domain, test input is selected among input domain for execution. Initially boundary value analyses are performed on each test cases. In boundary value analysis, the test is executed for first three input and last three input from input domain. For input domain InputD[0-11], the test case is executed for boundary value analysis for input 0, 1, 2 and 9, 10, 11.

2.3 Test cases execution

In the step, test cases are executed for testing. The input is randomly selected from the input domain which is generated using randomized algorithm.

2.4 Result comparison

During test cases execution result is generated for each test cases, these result compare with expected result. If the result is match with expected result then test case stored in executed test set, fail test cases are stored in fail result set.

3. Proposed Work

In proposed work, software random testing is performed using Monte Carlo and Las Vegas randomized algorithm. In figure 3 shows the basic flow for random testing, in which system is tested by using both random testing algorithms.

3.1 Create test plan

First we create test plan for testing which define detailed understanding of the eventual workflow. A test plan specifies the strategy that will be used to validate and make sure that a product or system meets its specifications and other requirements. A test plan is generally prepared by or with significant input from test expert persons.

3.2 Execute test cases using Monte Carlo Randomized algorithm

The test cases is generated using Monte Carlo randomized algorithm and executed that test cases. There are two possible result, execute and fail test cases. Execute test cases are stored in execute result set and fail test cases are stored in fail result set.

```
generateTestCases(inputDomain[first-1, last+1)
  begin
   i=0
   repeat
       Randomly select one element from input
       domain.
        i = i + 1
       until i=last-1  or error is found
  end
```

3.3 Execute test cases using Las Vegas Randomized algorithm

The test cases are generated using Las Vegas randomized algorithm and again executed fail result sets test cases. Sometimes Monte Carlo algorithm give false result so that we again test the fail result set to obtain better result.

```
generateTestCases(inputDomain[first-1, last+1)
begin
repeat
       Randomly select one element from input
       domain.
        until error is found
end
```

3.4 Generate report

In this step, we generate the report against fail test cases and executed test cases. Aim of this paper is to enhance the performance of random testing so that not concentrate on fails and pass test cases. We focus on how much time is taken to execute the test cases and comparison with previous methods.

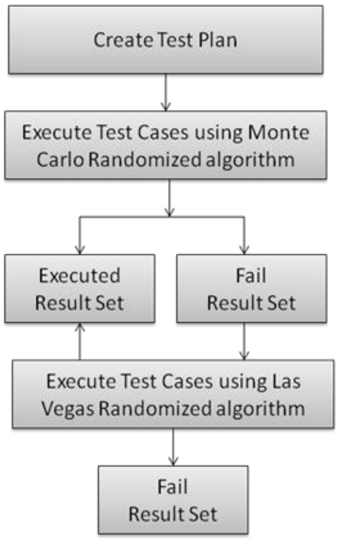

Create Test Plan
Execute Test Cases using Monte Carlo Randomized algorithm
Executed Result Set
Fail Result Set
Execute Test Cases using Las Vegas Randomized algorithm
Fail Result Set

Figure. 3. Proposed random testing implementations.

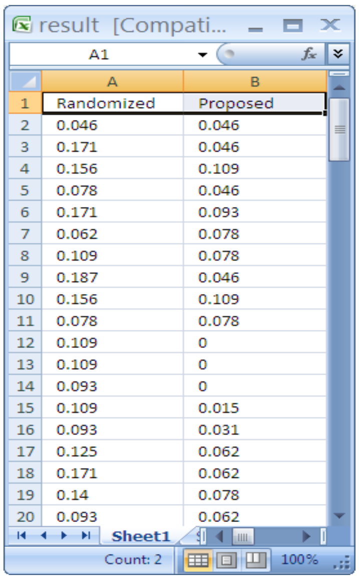
There are many methods for random testing, such as- Markov Chain Monte Carlo Methods [1], Adaptive Random Testing [2], [8], [9], [10], [17], Centroidal Voronoi Tessellations [3] , Dynamic Random Testing [4],[5] , Random-Partition Testing [4] [5], Combinatorial testing [14], Group Testing [15], Sneak paths testing [16]. Among these, adaptive random testing is more powerful and efficient. In this paper we compare proposed result with the adaptive random testing.

4. Result

In the below report shows time taken for executing test cases using randomized (adaptive) algorithm and our proposed algorithm. In this paper the test suit is downloaded from "NIST" website [19]. This website provides standard data for testing.

In the experiments, we have examined the performance of Random Testing (RT), Adaptive Random Testing (ART) and Dynamic Random Testing (DRT). As a result, combination of both algorithms is drastically improved against those of RT, ART and DRT. We present a technique that improves performance random testing.

We studied and implement random testing algorithm for enhancing performance of random testing. In proposed method uses Monte Carlo and Las Vegas randomized algorithm which is very helpful for test cases generation in different ways. Using proposed method, we can minimize the efforts for test cases generation process during random testing also minimizes testing efforts.



	A	B
1	Randomized	Proposed
2	0.046	0.046
3	0.171	0.046
4	0.156	0.109
5	0.078	0.046
6	0.171	0.093
7	0.062	0.078
8	0.109	0.078
9	0.187	0.046
10	0.156	0.109
11	0.078	0.078
12	0.109	0
13	0.109	0
14	0.093	0
15	0.109	0.015
16	0.093	0.031
17	0.125	0.062
18	0.171	0.062
19	0.14	0.078
20	0.093	0.062

Figure. 4. Time require executing all test cases using randomized and proposed technique.

In figure 5, shows the time require for executing all the test cases using random testing. In figure 6, shows the time require for executing all the test cases using proposed random testing. And in figure 7, shows the time comparison for both methods. Figure 7 shows that the time require for random proposed testing is less that time require for random testing, on this basis we can say that, proposed system is better than the previous one.

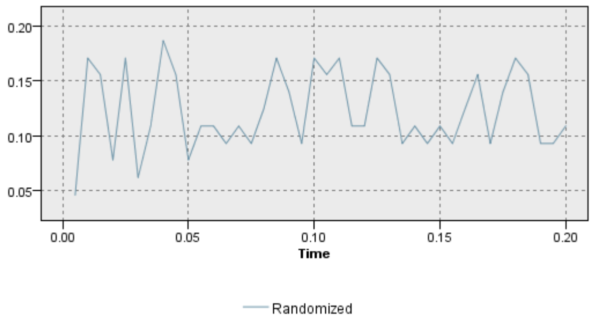


Figure. 5. Time require for random testing to execute all test cases.

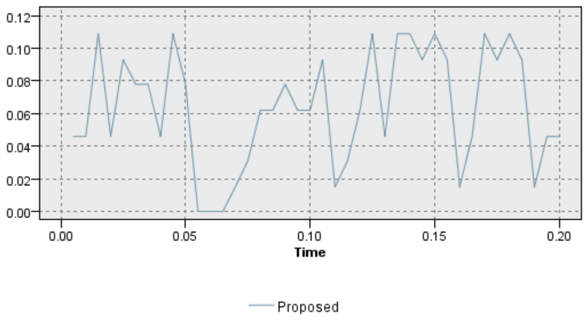


Figure. 6. Time require for proposed random testing to execute all test cases.

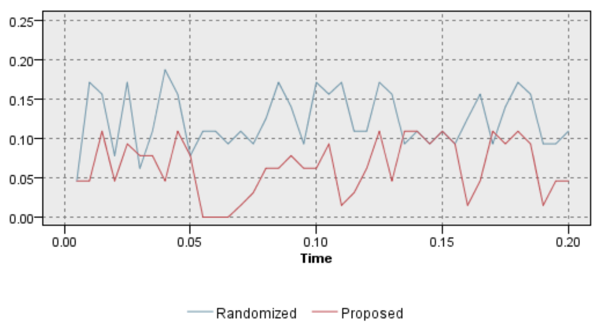
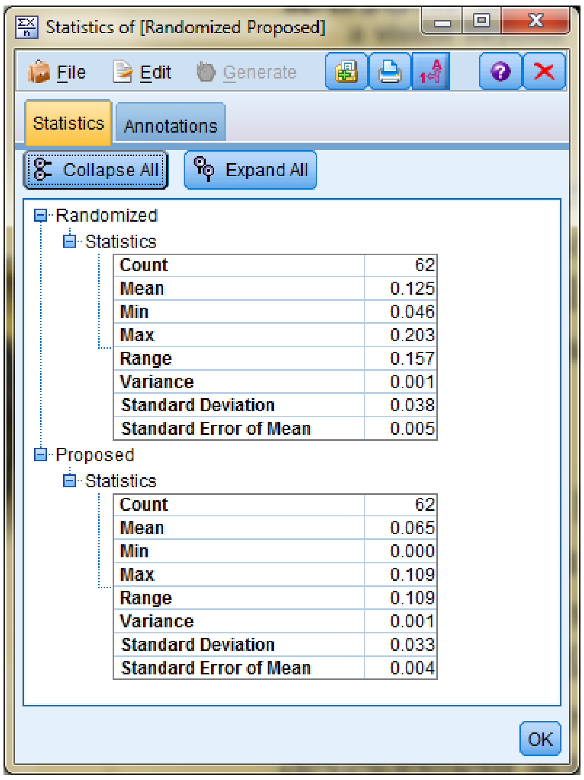


Figure. 7. Time comparison for both methods.

In figure 8, shows complete statistics report for random testing and proposed random testing. There are 62 test cases; the average execution time is 0.125 for random testing and 0.065 for proposed random testing. On the basis of this result we can say that our system is more effective and efficient than previous.



Randomized — Statistics

Count	62
Mean	0.125
Min	0.046
Max	0.203
Range	0.157
Variance	0.001
Standard Deviation	0.038
Standard Error of Mean	0.005

Proposed — Statistics

Count	62
Mean	0.065
Min	0.000
Max	0.109
Range	0.109
Variance	0.001
Standard Deviation	0.033
Standard Error of Mean	0.004

Figure. 8. Test cases execution statistics for both methods.

In this paper we presented the results that evaluate the performance of random testing. Using this method we can minimize testing efforts. Monte Carlo and Las Vegas algorithm is responsible for generating cases. We apply only one algorithm at a time according to code complexity. If complexity of code is high then use Las Vegas and for low complexity use Monte Carlo algorithm. The aim of the study was to observe how it performs random testing in general and to determine a more efficient strategy to recommend as best practice.

5. Features

- Minimize the efforts for test cases generation process
- Less calculation
- Simple procedure
- Consistency in test cases generation
- User profile
- Random input values
- Reliability
- Automated Testing

6. Conclusion

In this paper, we studied and implement random testing algorithm for enhancing performance of random testing. In proposed method uses Monte Carlo and Las Vegas randomized algorithm which is very helpful for test cases generation in different ways. Using proposed method, we can minimize the efforts for test cases generation process during random testing also minimizes testing efforts. Monte Carlo and Las Vegas algorithm is responsible for generating cases. The aim of the study was to enhance performs random testing. In the current scenario, Random testing is widely used for gaming and protocol testing because of running large number of test cases with high failure detection efficiency. Random testing cannot get perfect or optimal results, but it can get pretty good results with low cost. In some cases, random testing methods are more practical than any option.

7. Future Work

Randomized algorithms are not realistic, they are probabilistic. We should make it more realistic by applying various new approaches. In this testing there are many of the tests are redundant and unrealistic, we needs to remove redundant and unrealistic test cases. We also needs to minimize time which is spend on analyzing the test cases and facilitate to recreate the test if we do not record what data was used for testing.

8. References

[1] Bo Zhou, Hiroyuki Okamura, Tadashi Dohi "Enhancing Performance of Random Testing through Markov Chain Monte Carlo Methods" IEEE Transactions on Computers, VOL. 62, NO. 1, JANUARY 2013

[2] Tsong Yueh Chen, Fei-ChingKuo,HuaiLiu, W. Eric Wong "Code Coverage of Adaptive Random Testing" IEEE Transactions on Reliability, VOL. 62, NO. 1, MARCH 2013

[3] Ali Shahbazi, Andrew F. Tappenden, James Miller "Centroidal Voronoi Tessellations—A New Approach to Random Testing" IEEE Transactions on Software Engineering, VOL. 39, NO. 2, FEBRUARY 2013

[4] Junpeng Lv, Hai Hu, Kai-Yuan Cai "A Sufficient Condition for Parameters Estimation in Dynamic Random Testing" 2011 35th IEEE Annual Computer Software and Applications Conference Workshops, 978-0-7695-4459-5/11 $26.00 © 2011 IEEE

[5] Andrea Arcuri, Muhammad Zohaib Iqbal, Lionel Briand "Random Testing: Theoretical Results and Practical Implications" IEEE Transactions on Software Engineering, VOL. 38, NO. 2, MARCH/APRIL 2012

[6] Rajiv Chopra, Sushila Madan "Reusing Black Box Test Paths For White Box Testing of Websites" 978-1-4673-4529-3/12/$31.00 c-2012 IEEE

[7] Tsong Yueh Chen, Fei-Ching Kuo, and Huai Liu "Application of a Failure Driven Test Profile in Random Testing" IEEE Transactions on Reliability, VOL. 58, NO. 1, MARCH 2009

[8] Andrew F. Tappenden, James Miller "A Novel Evolutionary Approach for Adaptive Random Testing" IEEE Transactions on Reliability, VOL. 58, NO. 4, DECEMBER 2009

[9] Rubing Huang, Xiaodong Xie, Tsong Yueh Chen, Yansheng Lu "Adaptive Random Test Case Generation for Combinatorial

Testing" 2012 IEEE 36th International Conference on Computer Software and Applications, 0730-3157/12 $26.00 © 2012 IEEE

[10] Zhi Quan Zhou, Arnaldo Sinaga, Willy Susilo "On the Fault-Detection Capabilities of Adaptive Random Test Case Prioritization: Case Studies with Large Test Suites" 2012 45th Hawaii International Conference on System Sciences, 978-0-7695-4525-7/12 $26.00 © 2012 IEEE

[11] Padmanabhan Krishnan, R. Venkatesh, Prasad Bokil, Tukaram Muske, Vijay Suman "Effectiveness of Random Testing of Embedded Systems" 2012 45th Hawaii International Conference on System Sciences, 978-0-7695-4525-7/12 $26.00 © 2012 IEEE

[12] Manuel Oriol "Random testing: evaluation of a law describing the number of faults found" 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, 978-0-7695-4670-4/12 $26.00 © 2012 IEEE

[13] Bo SUN, Yunwei DONG, Hong YE "On Enhancing Adaptive Random Testing for AADL Model" 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing, 978-0-7695-4843-2/12 $26.00 © 2012 IEEE

[14] Andrea Arcuri, Lionel Briand "Formal Analysis of the Probability of Interaction Fault Detection Using Random Testing" IEEE Transactions on Software Engineering, VOL. 38, NO. 5, SEPTEMBER/OCTOBER 2012

[15] Marc Mézard, Cristina Toninelli "Group Testing With Random Pools: Optimal Two-Stage Algorithms" IEEE Transactions on Information Theory, VOL. 57, NO. 3, MARCH 2011

[16] Sachhidh Kannan, Jeyavijayan Rajendran, Ramesh Karri, Ozgur Sinanoglu, "Sneak-Path Testing of Crossbar-Based Nonvolatile Random Access Memories" IEEE Transactions on Nanotechnology, VOL. 12, NO. 3, MAY 2013

[17] Robert Merkel, Fei-Ching Kuo, Tsong Yueh Chen"An analysis of failure-based test profiles for random testing" 2011 35th IEEE Annual Computer Software and Applications Conference, 0730-3157/11 $26.00 © 2011 IEEE

[18] Rajeev Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, New York (NY), 1995

[19] http://samate.nist.gov/SRD/testsuite.php

Author Profile

Jaypal D. Rangari received B.Sc. degree in Mathematics from Nagpur University, Maharashtra, India. He received Master degree in Computer Application in 2012 from University of Mumbai. Currently pursuing M.Tech. (Final Year) in Computer Science and Engineering from G. H. Raisoni College of Engineering Nagpur, Maharashtra, India. His research area of interests in Software Engineering, Software Testing, Algorithm and Data Structure. At present he is engaged in “Enhance Performance of Random Testing using Randomized Algorithm” under the guidance of Prof. Swapnili P. Karmore