International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 4, April - 2014

A Comparative Study on Various Approaches to Anti Random Testing

Karthika Surendran[1]
Computer Science and Engineering
SCTCE
Trivandrum, Kerala, India

Syama. R[2]
Computer Science and Engineering
Assistant Professor
SCTCE
Trivandrum, Kerala, India

***Abstract*-The testing phase is of always primary concern to the programmers. It is not just because it takes maximum time, cost and effort but also because that it helps in identifying unrevealed errors in the software. Hence this area is of great interest to researchers. Over the years many techniques and approaches have been found to do the testing phase of the software. The two basic strategies of testing are white-box and black-box testing methods. This work mainly focuses on the five various anti random techniques, which is a black-box testing and an overview on the random testing is also made. This work is done as a comparative study of these five techniques. It has been recognized that the anti random testing can be applied to object-oriented software also. Hence a study on it is also done in the paper. The paper explains the different software failure input patterns and establishes the comparison work based on the study of the response of the testing methods to these patterns.**

Keywords- Anti random testing, various approaches to anti random testing, comparison of anti random testing techniques, a type of black-box testing

I. INTRODUCTION

Software testing is the phase in the software development process that aims at revealing all the unrevealed errors of the software so that these errors can be corrected. It is the last phase in the software development process. This phase mainly focuses on generating the test cases and also executing these test cases for validating the behaviour of the software system. Hence it is very much evident that quality of the test case can affect the quality of the software testing. This phase consumes maximum cost, effort and money during the software development process. The two basic methods of software testing are black-box and white-box testing. White-box testing, also called clear-box, glass-box, transparent-box and structural testing actually tests the internal structure and the working of the software. It is usually done at unit, integration, system levels of the process. Black-box testing, also called functional, behavioral, requirements and closed-box testing actually tests the functionality of the software. It is usually done at unit, integration, system and acceptance levels of the process. Random testing (RT) is a type of black-box testing which is efficient and is well used in industries. This technique is well explained in section III. However this technique has disadvantages.

Hence its improved technique called Anti Random Testing (ART) was developed over the years. The general ART algorithm is explained in detail in section III. This work mainly focuses on the ART. The various five approaches to Anti Random testing are studied in detail in this work. The five techniques of RT are explained in detail in section III. The first technique studied is Markov Chain Monte Carlo method (MCMC-RT) [1], which makes use of the Bayesian inference for generating random test cases. The second method discussed is Fixed Size Candidate Set (FCFS-ART) [2], which uses the distance between test cases and candidate set to find the next test case. The third method explained is Restricted Adaptive Random Testing by Random Partitioning (ART- RP Res) [3], which makes use of the principle of restriction to find the next test case. The fourth method is Adaptive Random Testing Based On Two Point Partitioning (ART-TPP) [4], which uses two points partitioning to generate the next test data. It is also found that ART can be used in the object-oriented cases also. The fifth technique is Feedback Directed Random Test Generation (RT- fd) [5], which can be applied to object-oriented programs. A comparison of these five techniques is shown in section IV. The comparison of the techniques is done based on the different types of inputs and the failure causing inputs which are explained in detail in section II. The work is concluded in section V with the inference obtained from the comparison made.

IJERT

II. SOFTWARE FAILURE PATTERNS

The inputs that are failure causing can be plotted together and it is found that this can form any of the three following patterns:-1) block failure pattern: the inputs that cause failure are within a specified area. This kind of block may be found in the statement block under a compound predicate according to the program code perspective. The example is for a fault in the branch such as *if(a<=x && x<=b && c<=y && y<=d)* , the failure causing input area can be denoted as *{(a,b),(c,d)}* (shown in fig.1), 2) strip failure pattern: the inputs may form the shape of a narrow strip pattern and may be predicate faults in a branch. For example if a programmer writes *if (x+y>=k2)* instead of *if(x+y>=k1)* the failure causing inputs will lie in a strip and the width can be calculated by $|k1-k2|$ (shown in fig.2) and 3) point pattern: the failure causing inputs may scatter into some points or small areas within the whole input domain. These faults can be found in branch with modulo operation or bitwise operation. For example some statements in a

branch *if(x%10==0 && y%10==0)* belongs to this category of failure causing inputs (shown in fig.3).

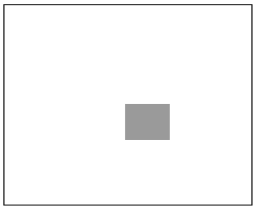


Fig.1. Block pattern

Fig.2. Strip pattern

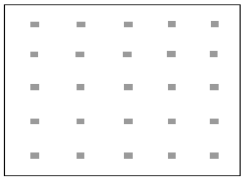


Fig.3. Point pattern

The percentage of the failure causing inputs is defined by failure rate and is denoted by θ. For a finite sized input domain with m failure causing inputs, $\theta = m/d$. The F - measure denotes the (random) number of test cases needed to detect the first failure. This is a natural measure for a testing strategy performance as it stops when the first failure is detected. The F-measure is used in all Adaptive Random Testingcases and hence it is ideal for comparison purposes. The theoretical mean F- measure can be calculated by $1/\theta$. For example, in the case of failure rate 0.01 the theoretical mean F- measure is equal to 100, which means that a failure will occur only after the execution of 100 test cases on average. The relative F- measure of a method is the F- measure of this method related to the theoretical mean F- measure of Random Testing, i.e. $1/\theta$. If the mean F- measure of a method is no worse than Random Testing then its relative mean F- measure should be below or atmost 1. The relative mean F- measure has the advantage that it is independent of failure rates and hence can be easily compared for different failure rates [3],[4],[12].

III. BACKGROUND

The random testing (RT) approach executes the testing process of the software by randomly generating independent test cases. The general RT technique can be explained with the algorithm that consists of the following steps[1]. In the first step, set an executed set to be empty. In the second step generate an initial test case by randomly selecting it from the input domain and then execute it. If no failure is revealed then add the executed test case to the empty set or else stop. In the third step generate new test cases based on the executed set of test cases. In the fourth step execute the test case generated in the third step and if no failure is revealed then add it to the set of test cases and perform the third step or else stop.

The random testing technique has various advantages like simplicity, high speed easiness in understanding, complete automation possible, cost effectiveness and so on. However applying it on the software some drawbacks are also been found. These drawbacks includes inefficiency, need for large test cases , using only the rate of failure causing inputs to compute the effectiveness [9], empirical knowledge is not addressed and so on.

Hence a new technique is developed and studied by Chen et al. [7] in 2004 to overcome the limitations of the random testing approach [6]. This technique is called the anti random testing (ART). It spreads the test cases to the maximum level possible. The executed set (set of distinct executed test cases that have not revealed any failures) and the candidate set (set of randomly selected test cases without replacement) are the two disjoint sets of test cases used in ART. Initially the executed set is taken empty. The first test case is generated by randomly selecting from the input domain. Until a failure is revealed the executed set will be incrementally updated with the chosen element from the candidate set. The farthest element in the candidate set from all the executed test cases is computed and it becomes the next test case [12]. Basically the ART can be again on the whole classified into categories: distance-based strategy and partitioning-based strategy [4]. The pseudo-code of ART is as follows:

Z={}
add random test case to Z and execute it
repeat until stopping criteria is satisfied
sample set $\boldsymbol{W}$ of random test cases
for each w of these |W| test cases
$w.minD = min(dist(w,z \in Z))$
execute and add to Z the w with maximum $minD$ [6]

IV. ANTI RANDOM TESTING APPROACHES

The anti random testing can be classified into various methods, based on the techniques used in each method to randomly generate the test cases. These methods are explained in this section.

A. MARKOV CHAIN MONTE CARLO METHOD (MCMC-RT)

The earlier MCMC-RT scheme had a disadvantage that it cannot be used for continuous inputs. Hence it could not be practically used with floating-point arguments that can be float and double. A new MCMC-RT technique was eventually developed from this method [1].

IJERT

The new method makes use of the distance between the points. The new method makes use of probability mass or density function to generate samples. The posterior distributions in Bayesian inference are effectively used to generate these samples. These samples are used in the method to handle the parameters. There are several sample generation techniques and one such effective technique is Metropolis Hastings (M-H) algorithm. The acceptance probability of the candidate and a random number determines whether the selected candidate should be added to the set of test cases or not. The steps of the MCMC-RT1 algorithm are as follows:

1. An initial test case is generated by random selection from the input domain.
2. A new candidate is generated according to the proposal distribution.
3. If the acceptance probability is less than a random number then the new candidate is not accepted as a test case or else the candidate is added to the set of generated test cases.
4. Steps 2 and 3 are repeated for a fixed number of iterations and the generated test case is returned as the result.

This algorithm takes large computation time because the steps 2 and 3 have to be repeated. This problem is solved in MCMC-RT2 by replacing the proposal distribution.

The drawback of the MCMC-RT1 is that steps 2 and 3 have to be repeated and this contributes to large computation time. To improve the situation the proposal distribution is replaced with the random walk process. The implementation of MCMC-RT2 is explained below.
The steps of the MCMC-RT2 algorithm are as follows:

1. A candidate is generated by random selection from the input domain.
2. If the acceptance probability is less than a random number step 1 is executed or else the generated candidate is accepted as the new test case.

MCMC-RT1 technique uses the iterative technique, which is a drawback of that method. Hence to improve the working the repetition of this technique is discarded and a new technique is hence developed, which forms the MCMC-RT2 method. MCMC-RT2 can be explained similar to MCMC-RT1 but without the repetition of steps. MCMC-RT2 technique is found to be faster as it does not involve the iteration step of M-H algorithm as in MCMC-RT1.

B. FIXED SIZE CANDIDATE SET ADAPTIVE RANDOM TESTING (FCFS-ART)

This is the first proposed ART method [2]. In this method a few candidates (one of which will eventually become the new test case) are randomly generated, say k candidates are generated randomly. For each candidate in the candidate list the test case that is previously executed and is closest to it is located and the distance between them are also calculated. The candidate with the largest such distance will become the next test case. This whole process will be repeated till the stopping condition is reached. The stopping condition is either the exhaustion of testing resources or the detection of potential failures.

The following figures explain the working of this algorithm. The four previously executed test cases are denoted by t_1 to t_4 and let c_1 to c_3 be the set of randomly generated candidates (value of k is taken as 3) from which one will be selected as the next test case. These test cases and candidates are shown in the fig.4. The fig.5 shows that the nearest previously executed test case of each candidate is found out and also that the distance between them is determined. These distances are compared in fig.6 and the candidate with largest such distance is selected as the next test case in fig.7. This process continues till the stopping condition is reached.

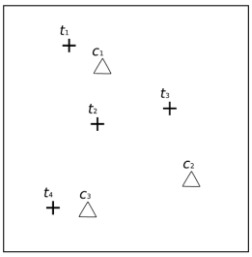


Fig.4. Multiple candidates are randomly selected

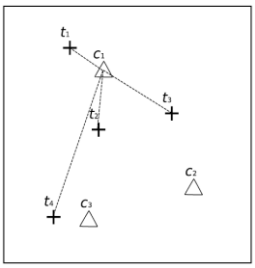


Fig.5. Nearest previously executed test case to each candidate is determined

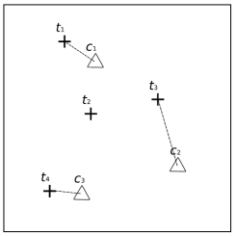


Fig.6. These nearest distances are compared along all candidates

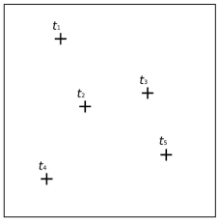


Fig.7. The candidate with longest such distance is selected

IJERT

C. RESTRICTED ADAPTIVE RANDOM TESTING(ART-RP Res)

The method of ART by Random Partitioning was proposed by Chen et al. [8]. In this method initially the whole input region is considered as the initial region and a test case is selected from it. Then the test case inclusive region is divided into four sub-regions. The next test case is randomly selected from the max-area region of these sub-regions. This process continues till the stopping condition is reached. This method is simple and have nearly linear runtime and with this method automation of Random Testing is also feasible and this method measures better F-value. But along with these advantages this method is also found to have a disadvantage that in all cases it cannot generate test cases widely and hence cannot avoid closely located test cases in all the situations.

Hence a new technique which includes the principle of restriction is developed [3]. The newly developed technique achieves minimum distance between the inputs by selecting test cases from restricted areas and hence avoids nearby and consecutive test cases. This method also requires only few test cases for its working.

The new method is similar to the old method developed by Chen but the difference is that this new method selects the next test case randomly only from the restricted version of the region with max-area. The region with the largest area is always sub-divided and among these sub-regions the region with maximum area becomes the next region from which the next case is randomly selected from its restricted version. This process continues till the stopping condition is reached. The stopping condition is either a failure is detected or the testing resources exhausts. The steps of the algorithm are as follows:

1. Initialize the candidate list of the test region with $\{\{(x_{min},y_{min})(x_{max},y_{max})\}\}$.
2. Select the test region with maximal area from the candidate list and remove it. If there are several such regions then one of them should be chosen randomly.
3. A point should be randomly selected from within the new restricted test region.
4. If the randomly selected point in step 3 is a failure-causing input then it should be reported and the process is to be terminated.
5. Else the current test region should be divided into four test regions and should be added to the candidate list.
6. Unless the resources for testing are exhausted proceed with step 2.

The following figures explain the working of this algorithm. Initially the method starts with the whole input region as the initial region (it is shown in fig.8). Fig.9 shows that the whole region is divided with respect to this point. Fig.10 and fig.11 shows that from among the sub regions the region with max-area is selected and it is from the restricted version of this sub region that the next test case is randomly selected. This process is continued until the stopping criterion has been reached.

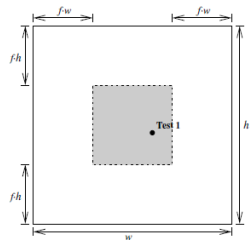


Fig.8. First test case selection

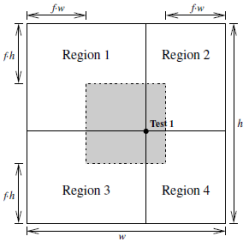


Fig.9. First partitioning

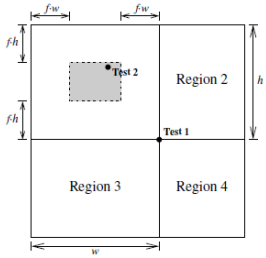


Fig.10. Second test case generation

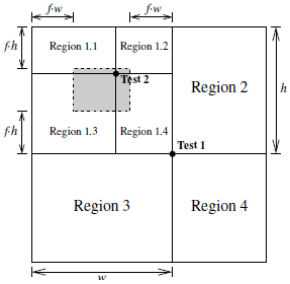


Fig.11. Second partitioning

However the key issue of this method is that it needs a test oracle, the software that can evaluate the outputs and decides “pass” or “fail”.

D. ADAPTIVE RANDOM TESTING BASED ON TWO-POINT PARTITIONING(ART-TPP)

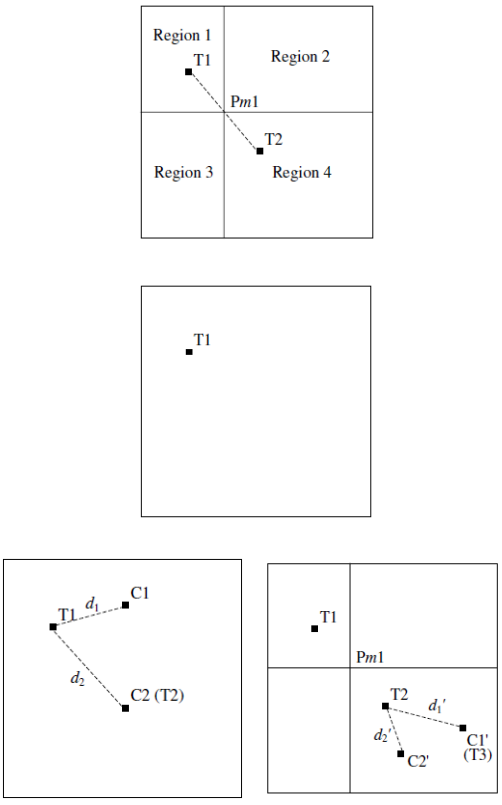
It is possible in the traditional adaptive random testing partitioning approaches for the two sampled test inputs to get very close to each other. Hence such a test case generation technique based on two points partitioning is proposed [4]. Basically the current region with the maximum area is selected as the object for partitioning and the partitioning is done at the midpoint of the two selected points. Initially a test point is randomly generated in the region. The second point is then selected from a set of candidates according to the criterion of farthest distance. This procedure of partitioning is

IJERT

repeated until the stopping condition is reached. The stopping condition is till either the potential faults are reached or the size of set of test data reaches a pre-set limit. The steps of the algorithm are as follows:

1. Add the whole input domain into the region list and initialize the set of data set as Φ.
2. Select the region with maximum area from the region list and remove it from the region list.
3. A new input point is to be randomly generated in this region if there are no previous test inputs in the max-area region and also it should be added to the set of data set or else step 4 is done.
4. Let the existing test input in the max-area region be denoted by T_i and generate k candidate points in the max-area region. Calculate the distance between T_i and each of the candidate point. Select the farthest point from T_i as the second test case (denoted as T_{i+1}) in the max-area region. Add this new test case to the set of test cases.
5. When a new test case is generated in the steps 3 and 4 and is added to the set of test cases it should be validated whether it can hit the failure-causing region or not. If it is true then the process is terminated or else the process is continued to append other test inputs.
6. The midpoint of the corresponding points of T_i and T_{i+1} is computed. The current max-area region is subdivided into four regions via this midpoint. These sub regions are then added to the list of regions. Continue with step 2.

Fig. 12 explains the operation of the algorithm in detail.



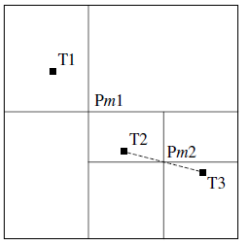


Fig.12. Illustration of two-point partitioning strategy

E. FEEDBACK DIRECTED RANDOM TEST GENERATION (RT-fd)

As the result of researches done in the anti random technique it was found this technique can be used for object-oriented programs also [5]. The algorithm uses earlier computed values as inputs to randomly select a method or constructor and hence generate test cases. This technique has got its name as it uses feedback from a test case to generate the next test case. If a test case is found to be error generating then it is unnecessary to build new test cases on it. Hence new test cases will be built only on test cases that are error free. The basic steps of the algorithm includes declaration of error sequences and non error sequences, creating new sequences, discarding duplicates, executing new sequences and checking contracts, classifying new sequences and outputs and if the condition is violated then it is error sequence else is non error sequence and apply filters. The technique includes steps such as: 1). Method sequences 2). Extending sequences 3).Feedback-directed generation 4).Filtering and 5).Repetition. The algorithm is as follows:

GenerateSequences**(***classes*, *contracts*, *filters*, *timeLimit***)**
1 *errorSeqs* ← {} *// Their execution violates a contract.*
2 *nonErrorSeqs* ←{} *// Their execution violates no contract.*
3 while *timeLimit* not reached do
4 *// Create new sequence.*
5 m(T_1 … T_k) ← *randomPublicMethod*(*classes*)
6‹*seqs*,*vals*›←*randomSeqsAndVals*(*nonErrorSeqs*; T_1 …T_k)
7 newSeq ← *extend*(m,*seqs*,*vals*)
8 *// Discard duplicates.*
9 if newSeq ε *nonErrorSeqs* U *errorSeqs* then
10 continue
11 end if
12 *// Execute new sequence and check contracts.*
13 ‹~o,*violated* › *execute*(newSeq,*contracts*)
14 *// Classify new sequence and outputs.*
15 if *violated* = *true* then
16 *errorSeqs errorSeqs* U {newSeq}
17 else
18 *nonErrorSeqs nonErrorSeqs* U {newSeq}
19 setExtensibleFlags (newSeq, *filters*; ~**o**) *// Apply filters.*
20 end if
21 end while
22 return ‹ *nonErrorSeqs*,*errorSeqs* › [5]

IJERT

V. COMPARISON

MCMC method can be concluded as a better option than ART and RRT; uses information on location of failures; has improved computation time and F-value; works also for discrete input; but has higher overhead value than RT.

ART-TPP is cost-benefit and has better performance and is best for block pattern but do not work in linear runtime. ART –RP Res is also simple, faster and effective and needs only lesser no: of test cases and works best for all the three patterns but works only for rectangular input domains and needs test oracle for test efficient execution.

RT-fd is a better method and can work without user inputs when implemented in RADOOP but only for Object Oriented Programming.

The results of the comparison done are shown in the table 1.

Table 1. Comparison table of output for each failure pattern

FAILURE PATTERN	INFERENCE
Point	ART-TPP is suitable
Strip	MCMC-RT is suitable and when θ is 0.005 ART-TPP is best and RP is worst and when θ is 0.01 TPP is worst
Block	ART-TPP is better for all values of θ and MCMC-RT is also suitable

VI. CONCLUSION

Testing is the last yet important phase of the software development process. The testing has two main strategies- black box testing and white box testing. This work mainly focuses on the various adaptive random testing techniques, which falls under the category of black-box testing. An overview of random testing is done initially. The general idea of adaptive random testing is explained. The work also includes a study on various failure input patterns to the software and the F-measure that can be used to express the performance of the different testing methods. The various adaptive random testing techniques are explained in the work. These techniques are not only studied but also compared according to their effectiveness of the performance. The application of adaptive random testing to object-oriented programs is also discussed in the work. The paper concludes

with the inference derived from the comparative study on these techniques.

VII. REFERENCES

[1] Bo Zhou, Hiroyuki Okamura, Tadashi Dohi, "Enhancing Performance of Random Testing through Markov Chain Monte Carlo Methods," IEEE Trans. Computers, vol. 62, no. 1, pp. 186-192, January 2013.

[2] Tsong Yueh Chen, Fei-Ching Kuo, Robert G. Merkel, T.H. Tse, "Adaptive Random Testing: the ART of Test Case Diversity," J. Systems and Software, vol. 83, no. 1, pp. 60-66, 2010.

[3] Johannes Mayer, "Restricted Adaptive Random Testing by Random Partitioning," 2012.

[4] Chengying Mao, "Adptive Random Testing Based on Two-Point Partitioning," Informatica 36, pp. 297-303, 2012.

[5] Carlos Pacheo, Shuvendu K. Lahiri, Michael D. Ernst, Thomas Ball, "Feedback-directed Random Test Generation," 2012.

[6] Andrea Arcuri, Lionel Briand, "Adaptive Random Testing: An Illusion of Effectiveness," 2012.

[7] T.Y. Chen, H. Leung, and I. K. Mak, Adaptive Random Testing In Advances in Computer Science, pp. 320-329, 2004.

[8] T. Y Chen, G. Reddy, R. Merkel, and P. K. Wong, "Adaptive random testing through dynamic partitioning," In Proceedings of the 4th International Conference on Quality Software (QSIC 2004), IEEE Computer Society, pp. 79-86, September 2004.

[9] http://link.springer.com/chapter/10.1007%2F978-3-540-30502-623#page-

[10] http://www.utdallas.edu/~ewong/SYSM-6310/03-Lecture/02-ART-paper-01.pdf

IJERT