



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

Test Suites Prioritization for Regression Testing using Genetic Algorithm

G.N Purohit, Emeritus Professor, Dept. of Mathematics & Computer Science, Banasthali University,
Banasthali(Raj.)

A.M Sherry, Professor & Director (IMT-CDL) Institute of Management & Technology, Ghaziabad, (U.P)
Manish Saraswat, Research Scholar, Banasthali University, Banasthali(Raj.)

Abstract— The applications of automated systems are increasing in our day to day life routine. These systems are developed by the human beings therefore some probability of fault in the product is likely to occur.

The testing is most important analytical method that examines the quality of system or software product by designing test cases and applying these in static or dynamic testing environment.

The quality assurance is an important concern for systems. The exhaustive testing is an idea in which system or software is tested with all possible test-cases and test- paths, but this consumes a lot of time and efforts to perform. Practically it is not possible to carry out exhaustive testing of systems; therefore a test-case or suite prioritization technique is required to assure quality of product without performing exhaustive testing.

In this paper a test case or suite prioritization technique has been proposed based on their capability to detect the errors in the program.

The test case prioritization technique is implemented using Genetic Algorithms which is based on optimization.

In this a test case sequence or suite is more likely to find the errors has been given a higher priority for execution and a test case sequence or suite which is less likely to find an errors is given a lower priority for execution

Index Words: Genetic Algorithm, Regression Testing, Test Cases.

I. INTRODUCTION

Software or systems are developed, tested and maintained by developers. There is always some probability of faulty software or systems. The execution/activities of faulty software or systems may produce in-correct or unpredictable output and this may be costly and even life threatening in critical applications which are highly paralleled, distributed, real time, embedded and implemented on host target environments. Testing is the one of most complicated and time consuming activity, it consumes approximately 50-60 % time and cost of the development life cycle [1]. Now a day's applications of automated systems is increasing in all aspects of our life, this also increases the complexity of systems or software, so that testing become more complicated and it is also required to cover technical characteristics like parallisms, distribution, timeliness, dynamic memory management and detection of dead lock. Broadly testing activities may be divided in two categories: Functional and Structural Testing. The functional testing concentrate on functional area of software, this testing assure that the software is working or doing all functions and operations as per specification, so that a functional testing confirm that a function's objective determined or specified by users or developer is correctly implemented. These testing techniques are also known as black box techniques, as no analysis is performed on the program code, only functions are listed in test specification document. The test-cases of functional testing are based on boundary value analysis, valid-invalid classes, distribution, worst cases analysis and decision table based testing. The structural testing is concerned with logical or internal structure of program. It's tried to find logical errors of program code thus in structural testing test cases are focused on code-coverage, path coverage, flow graph and statement traversal. Regression testing is another category of testing that assures that the modifications or changes introduced in program code does not cause additional defects or does not affect the correct definition of system. The regression testing is quite important for making sure that new modifications or changes do not affect the over-all functionality of software or system, to gain confidence about correctness of software or system a lot of testing is required, thus a lot of effort and effective time management is required. One straight forward approach is to re-run all existing test cases to assure that modifications in program do not cause any new defects, but practically it is not possible to re-test the software or system with all test cases and it is quite complicated and time consuming. The alternative approach is to prioritize the test-case sequence according to their error detection capability and execute the ordered sequence of test cases that contain the test cases first, which have more probability to find the errors in the product.

This paper is organized into four parts:

Part I: The various issues related to Software testing and it categories are included in this part and also focuses on the issues of the Regression Testing.

Part II This part discusses the existing research work related to prioritization techniques.

Part III This part explains the various related key features and implementation issues of GA for test case prioritization in regression testing.

Finally GAs is implemented to calculate fitness function value of various test cases and prioritize the test cases or suite execution order according to their errors detection capability.

Part IV concludes the paper.

II. RELATED WORK

A lot of research work is being carried out in the field of Test Data generation and test case prioritization, and various researchers proposed useful technique for test data generation, designing and prioritizing.

Ashraf *et al.*[5]proposed an algorithm which prioritizes the system test cases based on the six factors: customer priority, changes in requirement, implementation complexity, requirement traceability, execution time and fault impact of requirement. They conducted a controlled experiment on two industrial data sets to compare the proposed value based test case prioritization algorithm with random prioritization for early rate of fault detection and observed that algorithm based prioritization is more efficient to random prioritization. Sujata, and Kumar [2] presented a new approach for test case prioritization which is completely based on requirements of the system and this technique is also useful in black box environment. Ahmed *et al.*[4] proposed an approach for automating the test case prioritization process using genetic algorithm with Multi-Criteria Fitness function. They presented uses of multiple control flow coverage metrics. These metrics measure the degree of coverage of conditions, multiple conditions and statements that the test case covers. Rothermel *et al.*[6] describe several techniques for using test execution information to prioritize test cases for regression testing, including: techniques that order test cases based on their total coverage of code components, techniques that order test cases based on their coverage of code components not previously covered, and techniques that order test cases based on their estimated ability to reveal faults in the code components that they cover.

Huang *et al.* [3] has proposed a cost cognizant test case prioritization technique based on the use of historic records and genetic algorithm. They ran a controlled experiment to evaluate the proposed technique's effectiveness. Experimental results indicate that proposed technique frequently yields a higher Average Percentage of Faults Detected per Cost (APFDC).

Walcott [7] proposed a new test case prioritization technique using a genetic algorithm (GA). He used GA to prioritizes sub sequences of the original test suite so that the new suite, which is run within a time-constrained execution environment, will have a superior rate of fault detection when compared to rates of naively prioritized test suites.

III. RESEARCH KEY CONCEPTS

A. Genetic Algorithm

The genetic algorithm represents a class of adoptive search techniques and procedures are based on the process of natural genetics and Darwin's principle of survival of fittest. These algorithms model natural process such as Selection, Recombination, Mutation, Migration, locality and Neighborhood [8].

Genetic Algorithm is a search technique used to find exact or approximate solution of optimization and search problems. The GA use three basic operations: Selection, Re-Combination/Cross-Over and Mutation.

The Genetic Algorithm operates on strings of digits called chromosomes which are constructed by a digit that is called Gene. The collection of chromosome is called Population and the population is represented by a different code like binary, real number, permutation.

Each chromosome has an associated fitness and this value determine the probability survival of chromosome in the next generations.

After selection of chromosome for next generation a fixed percentage of chromosomes are crossed or re-combined [One point or multi point crossover] and a small percentage of genes are mutated. The algorithm for GA performed in following steps:-

- a. *Generate/Create a random population of chromosomes.*
- b. *Calculate/Evaluate the fitness [optimum] value of each member of population.*
- c. *Select the next generation population by allowing more chance of survival to chromosomes that have high fitness value.*
- d. *A Fixed percentage of population member selected for cross-over or Re-combination to create two new members.*
- e. *Perform a small percentage of mutation on Genes in created population for cross-over.*
- f. *Re-evaluate the fitness of chromosomes of population.*
- g. *Repeat the process till specified criteria is not met/Best population is constructed.*
- h. *End.*

B. Test Case and Test Suite

A test case is used to define the things that must be validated to ensure that the Software or system is working correctly and is built with a high level of quality. For instance, to test three different browsers, such as Firefox, Internet Explorer, and Safari, we can create three different test case execution records for each browser.

A test suite is a collection of test cases that are grouped for test execution purposes. A test suite may contain multiple test-cases.

C. Regression Testing

Software Testing is very challenging task and it is carried out throughout the SDLC to reveals the bugs. During the testing process bugs are corrected via modifying source code or definition so that it is necessary re-test the software or system, hence regression testing is evolved.

According to IEEE Regression Testing of System or Software or Component is to verify that modifications have not caused un-intended efforts and that the system or component still complies with its specified requirements [9] further regression testing may be Corrective (triggered by the changes in source code) and Progressive(triggered by specification changes).

In this paper our research is focused on test case and test suite selection during the regression testing, this is due to the fact that exhaustive testing is practically not possible and prioritized or careful selection of test cases can significantly reduce testing efforts and time.

D. Experiment Implementation

As discussed in previous section the genetic Algorithm is stochastic technique rather than deterministic, it is based on idea of selection of fittest chromosome and fitness of chromosome is defined by suitable objective function from entire population

The experiment implementation process is further divided in following steps:

D.1 Generation of Population [Chromosome]

Initially population is created by random selection and encoding, each chromosome represents the possible solution of the problem.

In our experiment the running sequence of test-cases is chromosome and we are optimizing this sequence of test cases execution to assure the quality of product after making modifications or changes. For Instance there are 15 test-cases [T₁, T₂, T₃,.....T₁₅] and two different test sequence of test cases identified to detect an error.

Test-Suite (TS₁) = T₁→T₂→T₃→T₄→T₆→T₈→T₁₀→T₁₂→T₁₃→T₁₁→T₁₄→T₁₅→T₅→T₇→T₉

Test-Suite (TS₂) = T₁₀→T₄→T₃→T₅→T₆→T₈→T₉→T₁₁→T₁₃→T₁₂→T₁₅→T₁₄→T₂→T₇→T₁

Fig 1: Two Test Suite (TS₁ & TS₂)

We will try to create best test sequence or test suite based on fitness of chromosomes.

D.2 Fitness Evaluation & Selection Criteria of Individual from Generated Population

The fitness of chromosome defined by objective function, this tells fitness of chromosome (possibility of survival) means it is bad or good. The objective functions generates a real number from the input chromosome, a chromosome may have higher or lower (function may be maximized or minimized) fitness value, prioritizes first in test case execution order and more probability of survival in next generation of population according to problem definition.

D.3 Crossover & Mutation

The selected individual are combined to generate new chromosomes, this technique for generating of chromosomes is called crossover. This may be of two types: single point or multi point crossover. For Example: two test suite or patents from fig 1, the test cases may be

C₁: T₁→T₂→T₃→T₄→T₆→T₈→T₁₀→T₁₂→T₁₃→T₁₁→T₁₅→T₁₄→T₅→T₇→T₉

C₂: T₁₀→T₄→T₃→T₅→T₆→T₈→T₉→T₁₁→T₁₂→T₇→T₁→T₂→T₁₃

For C₁ write first part of TS₁ and then second part of TS₂ with the constraint, which a test case has not been added in to C₁ and same process is followed to creation of C₂.

For doing mutation two genes are selected randomly swapped with each other, for example in C₁ genes T₁₀ and T₅ are selected and swapped as:

C₁: T₁→T₂→T₃→T₄→T₆→T₈→T₅→T₁₂→T₁₃→T₁₁→T₁₅→T₁₄→T₁₀→T₇→T₉.

D.4 Termination Criteria

The termination criteria can be set or selected in different ways such as: achieving the predefined fitness value, reaching on number of specified generation or non existing difference in the fitness values of each generation.

In this experiment we used a fixed number of generations as termination criteria.

E. Proposed Technique

Now we will discuss technique for test suite or test case prioritization using genetic algorithms.

Let's say a program(P) has test suite(TS),and now we are doing modifications in the program(P),suppose modified program is(P'),so that now to test modified program(P'),we will generate a prioritize sequence of test cases or test suites from the designed test-cases and test suites for the program(P').

The prioritization criterion is based on line of code modified by test case. In this paper following Genetic Parameter will be used:

E.1 Fitness Function: following objective function (fitness function) will be used to set prioritization of test case or test suite:

Fitness value (F) = $\sum \{order * (number \text{ of modified lines covered by test cases or suite})\}$

For example: A test case sequence is T1_T2_T3_T4 and this covers 3,2,5,1 modified lines of code respectively, then the fitness value of sequence of will be:

$$\text{Test Suite(TS)}(\text{fitness}) = (3*4) + (2*3) + (5*2) + (1*1) = 29.$$

E.2 Crossover: Here we have used one point cross over with crossover probability ($P_c=0.3$)

E.3 Mutation: In this paper we will use mutation probability $P_m=0.1$, it means that 10% of genes will be mutated within a chromosome.

Consider a simple program to classify a triangle. Its input is positive number (say a,b,c) and integer value of a,b,c must be greater than zero and less than or equal to 200. The program output may be one of following words: [Scalene, Isosceles, Equilateral, Not a triangle] [10].

The test engineer will be design various test cases or suite to classify triangles or to test the I/O parameters.

Assume that lines 5,7,9,12,16,19,23,37,31,35 of program are modified and assume that 15 test cases cover modified lines of code as shown in the table below:

Test-Case-ID	Number of modified lines covered by each Test-Case
T ₁	3
T ₂	1
T ₃	4
T ₄	2
T ₅	5
T ₆	2
T ₇	3
T ₈	6
T ₉	3
T ₁₀	5
T ₁₁	3
T ₁₂	2
T ₁₃	1
T ₁₄	0
T ₁₅	1

Table 1: Number of modified lines covered by each test-case.

Now we apply Genetic Algorithm, on this data as:

Chromosome(Test Sequence)	Fitness Value	Recommendation
T ₂ →T ₁ →T ₃ →T ₁₅ →T ₆ →T ₈ →T ₅ →T ₁₂ →T ₉ →T ₁₁ →T ₁₄ →T ₄ →T ₁₀ →T ₇ →T ₁₃	334	Chromosome-1
T ₁₀ →T ₄ →T ₃ →T ₅ →T ₆ →T ₈ →T ₉ →T ₁₁ →T ₁₃ →T ₁₂ →T ₁₅ →T ₁₄ →T ₂ →T ₇ →T ₁	396	Chromosome-2
T ₁ →T ₂ →T ₃ →T ₄ →T ₆ →T ₈ →T ₁₀ →T ₁₂ →T ₁₃ →T ₁₁ →T ₁₄ →T ₁₅ →T ₅ →T ₇ →T ₉	331	Chromosome-2

Table 2: Fitness value of three chromosomes (test case sequence) and recommendation

On the basis of this random number it is clear that first random number recommends the chromosome-1 that second random number recommends chromosome-2 and third random number also recommends chromosome-2 because of highest fitness value among three(more probability to survive or elite child for crossover for next generation chromosome).

So now we have following member in our mating pool:

C₁: T₂→T₁→T₃→T₁₅→T₆→T₈→T₅→T₁₂→T₉→T₁₁→T₁₄→T₄→T₁₀→T₇→T₁₃

C₂: T₁₀→T₄→T₃→T₅→T₆→T₈→T₉→T₁₁→T₁₃→T₁₂→T₁₅→T₁₄→T₂→T₇→T₁

C₂: T₁₀→T₄→T₃→T₅→T₆→T₈→T₉→T₁₁→T₁₃→T₁₂→T₁₅→T₁₄→T₂→T₇→T₁

Now we apply one point crossover on these chromosomes and that will generate new off springs or Recombinated chromosomes (RC) as:

RC₁: T₂→T₁→T₃→T₁₅→T₆→T₈→T₅→T₁₁→T₁₃→T₁₂→T₁₄→T₄→T₁₀→T₇→T₁₃

RC₂: T₁₀→T₄→T₃→T₅→T₆→T₈→T₉→T₁₂→T₁₃→T₁₁→T₁₄→T₂→T₇→T₁→T₁₅

RC₃: T₁₀→T₄→T₃→T₅→T₆→T₈→T₉→T₁₂→T₁₃→T₁₁→T₁₄→T₂→T₇→T₁→T₁₅

T₂→T₇→T₁→T₁₅

T₂→T₇→T₁→T₁₅

Now suppose crossover probability is .3 then we select two chromosomes from off spring and one chromosome from the parents based on fitness function value for the next generation and mutation probability is .1, this will change 10% of genes of new off springs for the next generation.

This process is continued for a fixed number of generations and after repeating of multiple times, there is a high probability for getting nearly optimal solution.

The simulation based modeling is useful to represent key characteristics and behavior of functions of selected systems.

IV. CONCLUSION

In this paper we applied genetic algorithm on test cases to prioritize their execution during Regression testing of system or software. We used a fitness function to determine efficiency of test case(a test case cover more number modified lines is more efficient) and a test case sequence(or test suite) which has higher fitness value have higher priority for execution during the testing. On applying genetic algorithm for a large number of time or generations, there is higher probability that we will get nearly optimum solution.

REFERENES

- [1] Hetzel, B: Complete guide of Software Testing Wellesley, MA: QED Information Sciences.
- [2] Sujata, Varun Kumar: "Requirements based Test Case Prioritization using Genetic Algorithm", IJCST, Vol-3, Dec2010.
- [3] Yu-Chi Huang, Jun-Ru Chang and Tsan-Yuan Chen "Design and Analysis of Cost-Cognizant Test Case Prioritization Using Genetic Algorithm with Test History", IEEE 34th Annual Computer Software and Applications Conference 2010.
- [4] Amr abdelfatah Ahmed Dr. Mohamed Shaheen, Dr. Essam Kosba: "Software testing suite Prioritization using multicriteria Fitness function" ICCTA 2012, 13-15 October 2012, Alexandria, Egypt
- [5] E. Ashraf, A. Rauf, and K. Mahmood: "Value based Regression Test Case Prioritization" Proceedings of the World Congress on Engineering and Computer Science 2012 Vol I.WCECS 2012, October 24-26, 2012, San Francisco, USA.
- [6] Gregg Rothermel: "Prioritizing Test Cases for Regression Testing" IEEE Transactions on Software Engineering, Vol. 27, no. 10, October 2001.
- [7] Kristen Walcott: Prioritizing Regression Test Suites for Time-Constrained Execution Using a Genetic Algorithm, Technical Report CS05-11, Submitted to the Faculty of the Department of Computer Science, Allegheny College 2005.
- [8] Pohleim,H: GEA Tbx. Genetic and Evolutionary Algorithm Tool Box for use with Matlab Documentation ,Technical Report,1996.
- [9] L.Granja& M. Jino: "Technique for Regression testing: Selecting test case sets tailored to possibly modified function, In proceedings of third European conference, page-2-11, 1999.
- [10] K.K. Agarwal, Yogesh Singh, "Software Engineering",New AgeInternational Publishers,India 2006.
- [11] Ruchi Malhotra et.al. Test Case Prioritization using Genetic Algorithm, Delhi Technological University New Delhi