January 2014

TEST CASE PRIORTIZATION USING GENETIC ALGORITHM

RUCHIKA MALHOTRA

Computer of engineering department, Delhi Technological University, Ruchikamalhotra2004@yahoo.com

ABHISHEK BHARADWAJ

Computer of engineering department, Delhi Technological University, abhishek.bharadwaj@dtu.co.in

Follow this and additional works at: <https://www.interscience.in/ijcsi>

Part of the [Computer Engineering Commons](#), [Information Security Commons](#), and the [Systems and Communications Commons](#)

Recommended Citation

MALHOTRA, RUCHIKA and BHARADWAJ, ABHISHEK (2014) "TEST CASE PRIORTIZATION USING GENETIC ALGORITHM," *International Journal of Computer Science and Informatics*: Vol. 3 : Iss. 3 , Article 1.

DOI: 10.47893/IJCSI.2014.1136

Available at: <https://www.interscience.in/ijcsi/vol3/iss3/1>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

TEST CASE PRIORTIZATION USING GENETIC ALGORITHM

RUCHIKA MALHOTRA¹ & ABHISHEK BHARADWAJ²

^{1,2}Computer of engineering department, Delhi Technological University,
(Formerly Delhi College of engineering) New Delhi, India

Email: Ruchikamalhotra2004@yahoo.com, abhishek.bharadwaj@dtu.co.in

Abstract: Software is built by human so it cannot be perfect. So in order to make sure that developed software does not do any unintended thing we have to test every software before launching it in the operational world. Software testing is the major part of software development lifecycle. Testing involves identifying the test cases which can find the errors in the program. Exhaustive testing is not a good idea to follow. It is very difficult and time consuming to perform. In this paper a technique has been proposed to do prioritize test cases according to their capability of finding errors. One which is more likely to find the errors has been assigned a higher priority and the one which is less likely to find the errors in the program has been assigned low priority. It is recommended to execute the test cases according their priority to find the errors.

Keywords: Genetic algorithm; testcase priortization, test case minimization

1. INTRODUCTION

Software testing is the process of executing the program with the intent of finding errors. [1]

When we test the software in the maintenance phase after the change has been incorporate, this is called regression testing. So regression testing is also quite important for making sure that the new modifications do not add any extra faults. This regression testing requires lots of effort and time.

One straight forward approach is to re-run all the existing test cases and detect if there are any errors. But it is practically impossible under the project deadline and required a lot of effort. Other alternative is to do prioritize test cases according to their relevance for error detection and find an ordered sequence of test cases which contains the test cases first, which is more likely to find errors.

Testing activity can be defined in two broad categories-

- a. Functional testing
- b. Structural testing

Functional testing includes the functional part of the software. It is used to assure that the software do what it is expected to do. It includes the following testing approaches-

1. Boundary value analysis
2. Robustness testing
3. Worst case testing
4. Equivalence class testing
5. Decision table based testing
6. Cause effect graph testing

Structural testing deals with the internal structure of the programs. It concern with the code of the program. It include the following approach-

1. Path testing
2. Flow graph testing
3. DD path graph testing
4. Data flow graph testing

In this paper we have proposed a technique to order the test cases according to their priority to find

faults. A test case which is more likely to find an error will be given more priority and hence kept first in the ordered sequence and so on. This order will be generated using genetic algorithm.

2. RELATED WORK

In various research work carried out in the field of test data generation, different researchers have used different technique while generating test data.

Yu-Chi Huang et al has proposed a cost cognizant test case prioritization technique based on the use of historic records and genetic algorithm [2]. They run a controlled experiment to evaluate the proposed technique's effectiveness. This technique however does not take care of the test cases similarity.

Sangeeta Sabharwal et al has proposed a technique for prioritization test case scenarios derived from activity diagram using the concept of basic information flow matrix and genetic algorithm.[3]

Sangeeta Sabharwal et al has generated prioritized test case in static testing using genetic algorithm.[4] they have applied a similar approach as [3] to prioritize test case scenarios derived from source code in static testing.

James H. Andrews et al has applied genetic algorithm on randomized unit testing to figure out the best suitable test cases.[5]

Mohsen Fallah Rad et al has applied common genetic and bacteriological algorithm for optimizing testing data in mutation testing.[6]

Ruchika Malhotra et al has developed a adequacy based test data generation technique using genetic algorithms.[7]

Ciyong Chen et al proposed a new method called EPDG-GA which utilizes the Edge Partitions Dominator Graph (EPDG) and Genetic Algorithm (GA) for branch coverage testing.[8]

Dr Mukesh kumar, rohit et al has proposed unit testing of object oriented software using genetic

algorithm. In their approach they proposed a method to generate the test cases for classes in object oriented software using a genetic programming approach. This method represents a tree representation of statements in the test cases. Strategies for encoding the test cases and using the objective function to evolve them as suitable test cases are proposed.[9]

Debasis Mohapatra et al has proposed automated test case generation and its optimization for path testing using genetic algorithm and sampling. In this approach they have used genetic algorithm to optimize the test cases that are generated using the category- partition and test harness pattern.[10]

Md. Imrul Kayes proposed test case prioritization for regression testing based on fault dependency[11]. He present a metric APFDD which measure fault dependency detection rate and presented an algorithm to improve APFDD.

Zheng Li et al have applied search algorithm for regression test case prioritization.[12]

Gregg Rothermel et al have performed a control experiment to access prioritization techniques using mutation faults.[13]

Gregg Rothermel et al have proposed several techniques for developing prioritize test cases in regression testing phase. They also rate of fault detection of these techniques.[14]

3. KEY RESEARCH CONCEPT

3.1 Genetic algorithm

Genetic algorithm is stochastic search technique, which is based on the idea of selection of the fittest chromosome.

In genetic algorithm, population of chromosome is represented by different codes such as binary, real number, permutation etc. genetic operators(i.e. selection, crossover, mutation) is applied on the chromosome in order to find more fittest chromosome. The fitness of a chromosome is defined by a suitable objective function. As a class of stochastic method genetic algorithm is different from a random search. While genetic algorithm carry out a multidimensional search by maintaining population of potential user, random methods consisting of a combination of iterative search methods and simple random search methods can find a solution for a given problem. One of the genetic method's most attractive feature is to explore the search space by considering the entire population of the chromosome.[15]

The steps of genetic algorithm are as-

1. Generate population (chromosome)
2. Evaluate the fitness of generated population
3. Apply selection for individual
4. Apply crossover and mutation
5. Evaluate and reproduce the chromosome

3.2 Generate population(chromosome)-

Initially population is randomly selected and encoded. Each chromosome represent the possible

solution of the problem.(in our case the sequence of test cases is chromosome and our aim is to optimize this sequence). For example- for 12 test cases T1, T2, T3.....T12 the sequence is T1→T2→T4→T6→T9→T10→T12→T3→T5→T7→T8→T11

3.2.1 Evaluate the fitness of generated population-

The fitness of a chromosome is defined by an objective function. An objective function tells how 'good' or 'bad' a chromosome is. This objective function generates a real number from the input chromosome. Based on this number two or more chromosome can be compared.

3.2.2 Apply selection for individual-

In general the selection is depend on the fitness value of the chromosome. The chromosome with higher or lower value will be selected base on the problem definition.

3.2.3 Apply crossover and mutation-

Parents are choose and randomly combined. This technique for generating random chromosome is called crossover. There exist two type of crossover-

- a. Single point crossover
- b. Multiple point crossover

For example- suppose two sequences for test cases is

P1: T1→T2→T3→T4→T5→T6→T7→T8→T9

and

P2: T4→T2→T5→T7→T8→T1→T6→T9→T2

Then using one point crossover offspring will be-

C1: T1→T2→T3→T4→T8→T6→T9→T5→T7

C2: T4→T3→T5→T7→T6→T8→T9→T1→T2

For C1 write first part of the P1 as it is and then write second part of P2 with constraint that a test case has not been added in to C1.

For doing mutation two genes selected randomly along the chromosome and swapped with each other.

For example- when T3 and T9 get selected randomly

T1→T2→T3→T4→T8→T6→T9→T5→T7

T1→T2→T9→T4→T8→T6→T3→T5→T7

3.2.4 Termination criteria-

The termination criteria can be selected in the different ways such as- reaching the predefined fitness value, the number of generation or a non-existing difference in the fitness values of each generation.

In our approach we used a fixed generation number as a termination criteria.

Figure 1: steps in genetic algorithm [11]

4. PROPOSED TECHNIQUE

In this section we present technique for test case prioritization using genetic algorithm.

Let's say a program has test case suite T, now if we make modification in the program p, suppose modified program is P', so in order to test program P' we will generate a prioritize sequence of test cases from test case suite T, on the basis of the line of code modified.

In this paper the following genetic parameter will be used-

1. **Fitness function** : The following objective function (fitness function) will be used-

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

For example- a test case sequence is T1_T2_T3_T4 and T1, T2, T3 and T4 covers 2,1,5,3 modified lines of code respectively. Then fitness value for this sequence will be

$$F = (2*4) + (1*3) + (5*2) + (3*1) = 16$$

In this T1 has order 4 and it covers 2 lines of code, T2 has order 3 and it contains 1 line of code, T3 has order 2 and it covers 5 line of code and T4 has order 1 and it covers 3 lines of code.

2. **Crossover** : In this proposed paper we will use one point cross over with crossover probability $P_c=0.33$.

3. **Mutation** : In this paper we will use mutation probability $P_m=0.2$. it means that 20% of the genes will be mutated within a chromosome.

Example -Test cases with execution history [1].

Test case ID	A	B	C	Expected Output	Execution History
T1	30	20	40	Obtuse angled triangle	8, 9, 10, 11, 12, 13
T2	30	20	40	Obtuse angled triangle	8, 9, 10, 11, 12, 13, 14, 15, 16, 20, 21, 22
T3	30	20	40	Obtuse angled triangle	10, 11, 12, 13
T4	30	20	40	Obtuse angled triangle	10, 11, 12, 13, 14, 15, 16, 20, 21, 22

T5	30	20	40	Obtuse angled triangle	12, 13, 14, 15, 16, 20, 21, 22
T6	30	40	50		22, 23, 24, 25, 28
T7	30	20	40	Obtuse angled triangle	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 20, 21, 15, 16, 20, 21, 35
T8	-	-	-	-	
T9	30	10	15		5, 6, 7, 8, 9, 10, 11, 12, 14, 17, 18, 19, 20, 21
T10	30	10	15		18, 19, 20, 21, 35
T11	30	20	40	Obtuse angled triangle	24, 25
T12	30	20	40	Obtuse angled triangle	15, 16, 20, 21

Assume that lines 5, 8, 10, 15, 20, 23, 28, 35 are modified and the modified lines of code covered by each test case are shown in the table below-

Test case	Number of modified lines
T1	2
T2	4
T3	1
T4	3
T5	2
T6	2
T7	5
T8	2
T9	4
T10	1
T11	0
T12	2

Table 2: number of modified lines covered by test case

Now we apply genetic algorithm, on this data.

Chromosome	Fitness value	Normalized value	Cumulative probability	Selection of random number	Recommendation
T1→T2→T3→T4→T5→T6→T7→T8→T9→T10→T11→T12	196	196/573=0.342	0.342	0.3	Chromosome 1
T2→T4→T6→T8→T10→T12→T1→T3→T5→T7→T9→T11	189	189/573=0.329	0.671	0.4	Chromosome 2
T5→T6→T	188	188/5	1	0.2	Chr

8→T9→T12 →T1→T7→ T11→T2→ T3→T4→T 10		73=0. 328			omo som e 1
---	--	--------------	--	--	-------------------

On the basis of this random number we got to know that the first random no recommends the chromosome 1 that is (T1→T2→T3→T4→T5→T6→T7→T8→T9→T10→T11→T12) because the selected random no lies between 0-0.342. Second random number recommends the chromosome 2 that is (T2→T4→T6→T8→T10→T12→T1→T3→T5→T7→T9→T11) because the random number lies between 0.342-0.671. The third random number recommends the chromosome 1 (T1→T2→T3→T4→T5→T6→T7→T8→T9→T10→T11→T12) because the selected random number lies between 0-0.342.

So now we have the following member in our mating pool:

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

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

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

Now we will apply the one point cross over on these chromosome and will generate the new off springs.

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

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

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

On applying one point cross over the selected population we will get the following off springs-

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

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

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

Now suppose if the crossover probability is 0.3 then we select 2 chromosomes from the offspring and one from the parents based on the fitness function value.

This process is repeated certain fixed number of iterations, on repeating this procedure multiple times, we will get the nearly optimum solution.

5. CONCLUSION

In this paper we applied genetic algorithm on the test cases with their execution history. We used a fitness function which gives higher value if a test case covers more line of code, and a test case which has higher fitness value is provide higher priority in ordered sequence. When we applied genetic

algorithm a large number of time we will get a nearly optimized solution. As we know that genetic algorithm does not always gives optimum solution, but if we run this algorithm fairly large number of time then we will get nearly optimum solution.\

REFERENCES

1. K.K. Aggarwal, Y. Singh, "Software Engineering", New Age International Publishers, 2006.
2. Yu-Chi Huang, Chin-Yu 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.
3. Sangeeta Sabharwal, Ritu Sibal, Chayanika Sharma "Prioritization of Test Case Scenarios Derived from Activity Diagram Using Genetic Algorithm", international conference. on Computer & Communication Technology[ICCT 10].
4. Sangeeta Sabharwal, Ritu Sibal, Chayanika Sharma "A Genetic Algorithm Based Approach For Prioritization of Test Case Scenarios in Static Testing ", International Conference on Computer & Communication Technology (ICCT)-2011.
5. James H. Andrews, Member, IEEE, Tim Menzies, Member, IEEE, and Felix C.H. Li "Genetic Algorithms for Randomized Unit Testing", IEEE transaction on software Engineering, vol. 37, no. 1, January/February 2011.
6. Mohsen Fallah Rad, Farshad Akbari Ahmad Javan Bakht "Implementation of Common Genetic and Bacteriological Algorithms in Optimizing Testing Data in Mutation Testing".
7. Ruchika malhotra and mohit garg "An Adequacy Based Test Data Generation Technique Using Genetic Algorithms" Journal of Information Processing Systems, Vol.7, No.2, June 2011.
8. Ciyong Chen , Xiaofeng Xu , Yan Chen , Xiaochao Li and Donghui Guo "A New Method of Test Data Generation for Branch Coverage in Software Testing Based on EPDG and Genetic Algorithm".
9. Nirmal Kumar Gupta and Dr. Mukesh Kumar Rohil "Using Genetic Algorithm for Unit Testing of Object Oriented Software", in First International Conference on Emerging Trends in Engineering and Technology.
10. Debasis Mohapatra Prachet Bhuyan Durga P. Mohapatra "Automated Test Case Generation and Its Optimization for Path Testing Using Genetic Algorithm and Sampling", 2009 WASE International Conference on Information Engineering.
11. Md. Imrul Kayes "Test Case Prioritization for Regression Testing Based On Fault Dependency".
12. Zheng Li, Mark Harman and Robert M. Hierons "Search Algorithm For Regression Test Case Prioritization".
13. Hyunsook Do and Gregg Rothermel "A controlled Experiment Assessing Test Case Prioritization via Mutation Faults".

◆◆◆