

A Survey on Test Case Prioritization Techniques

Geetha.U, Dr.Sharmila Sankar , Dr.Sandhya

(Dept of Information and Technology, Dept of Computer Science and Engg, Dept of Computer Science and Engg)

B.S.Abdur Rahman Institute of Science and Technology, Chennai, India

(geetha_it_phd_17@bsauniv.ac.in, sharmilasankar@bsauniv.ac.in, sandhya@bsauniv.ac.in)

Abstract:

Software testing is one of the most important phases in software development life cycle .In STLC, regression testing is one of the most expensive process that determines whether any new problem are the result of software modification or changes. Before making changes, software is tested with larger number of test cases and then successfully delivered .But change is applied, testing a code with all test cases is too expensive and time consuming process. So either whole software should be tested with few test case or in selected area with few test case to detect the fault or new issue due to changes in the existing code .Due to this ,effective test case prioritization techniques are required to find effective test cases. There many conventional method used for solving this problem but most of the excising techniques are facing the problem with coverage ratio. So this paper gives overview of different prioritization techniques. Finally this paper gives detailed survey of greedy and genetic based prioritization techniques and concluded that genetic algorithm is more efficient for test case prioritization.

Keyword: Regression testing, test case generation, test case optimization, test case prioritization.

1. Introduction

Software testing is the process of testing the quality of the product or software under test and providing the relevant information about the quality to its stakeholders. Software Testing eases out the process of understanding risks associated with the implementation of software and gives a detailed idea on which part of the software is to be worked on in order to get the required output. Regression testing is a type of software testing which is done to check whether the software gives desired output even after changes have been made to it. Changes may include software enhancements, patches, configuration changes, etc. During regression testing, new software bugs or faults can be found. The purpose of regression testing is to ensure that changes such as those mentioned above have not introduced new faults. One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software. Software Regression Testing is carried out to reduce the efforts of testing the software by selecting only the required numbers of test cases in order to test the software after changes have been made to it. This ensures that the fault detection rate is increased.

1.1 Test Case prioritization Techniques

Test case reduction techniques are classified as follows: Coverage based, Requirement based, Slicing, Greedy Algorithm, Genetic Algorithm, Hybrid Algorithm, Fuzzy Logic and Clustering.

Genetic Algorithm:

A genetic algorithm (GA) is a natural selection based process that is similar to biological evolution and a widely used method for solving both constrained and unconstrained optimization problems. In every iteration, the genetic algorithm selects parents from the current population and produces children for the next generation. Over successive generations, the population tends to an optimal solution. In the beginning, we have initial population and we define fitness value for every individual in the population. Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

Requirement Based:

The ultimate aim of test case reduction is to optimize the test case suite for regression testing. This optimization can be performed in many ways. One such way to optimize the test case suite is to minimize the testing requirements. This is known as Requirement based Test Case Reduction Technique.

Fuzzy Logic:

Optimization of test suites can also be achieved by using fuzzy logic. The fuzzy logic testing is based on an objective function. This technique provides the safest optimal solutions.

Clustering:

The Clustering technique focuses on removal of redundant test cases in the test case suite thereby, reducing the number of test cases to be executed. This is done using data mining methods for clustering. The result is smaller representative sets of test cases without redundancy. But they will have less fault detection ability.

Coverage Based:

The coverage based technique ensures that the majority of execution paths in a program are exercised. This is done using Control flow graphs which can be derived from a given program. Each state in this graph can reveal a fault.

Program Slicing:

Program slicing technique is used to check a program over a specific property and to build a slice set. There are three types of slicing techniques: dynamic slicing, static slicing, and relevant slicing. This method focuses on parts of the program code that have significant impact on its output. Whereas, those parts of program that have no effect on the program output are eliminated from testing process. Hence, as the size of the code gets reduced, testing time and cost will also be reduced.

Greedy Algorithm:

Greedy algorithm considers test cases as objects and requirements as attributes. One of well-known heuristics proposed for code-based reduction is Greedy algorithm. Initially, a context table is constructed. This table is worked on by reducing the objects and attributes which reduces the size of the table. The reduction of objects and attributes is based on redundancy. The redundant attributes and objects are not considered into the next iteration.

Hybrid Algorithm:

Hybrid algorithm takes into consideration, the usage of more than one test case reduction Techniques. For example, the combination of genetic algorithm with greedy approach can be used for good results. Various optimization objectives are also considered together.

2. Test case Prioritization techniques

2.1 Greedy Approach

Greedy approach uses coverage information of each test case and test cases are selected which one has higher coverage information and then rest the other test cases for lower priority.

I. Algorithm

```

1.Ti={1..n} //test suite
2. Si={1..n} //set of statements
3.for every Test case in Ti
  for every statement in Si
    if Si covered by Ti
      covSi=1;
    else
      covSi =0;
  End for
End for
4.Find weight map for all test case and select test case which covering more
statement.
5. Find statements covered by selected test cases and Mark '0'.
6. Mark "0" in the other test case wherever statements which are covered by
selected one.
7. Repeat step through step 3 to 6 until all statements are covered.

```

Algorithm 3

II. Case study

The following program has been considered for test case reduction using Greedy algorithm

Greedyex(a, b, c){	STATEMENT
If (a<b) {	S1
If(a==c){	S2
Print("cresent")	-
}}	-
Else If(b>c){	S3
If(b==a){	S4
Print("university");}}	-
Else{	-
c=15;	S5
Print("chennai");	-
}}	-

Program 3

III. Initial test cases and reduced test cases

Initially there are four test cases in the original test suite .The original test suite follows

Test case	A	B	C
T1	4	4	3
T2	2	7	1
T3	2	2	16
T4	1	8	5

Table 7

Coverage information for all test cases follows

Test cases	S1	S2	S3	S4	S5
T1	0	0	1	1	0
T2	1	0	0	0	0
T3	0	0	0	0	1
T4	0	0	1	0	0

Table 8

Test case	Statements covered
T1	3,4
T2	1
T3	5
T4	3

Table 9

Weight mapping: 2,1,1,1

Now greedy algorithm selects the test case T1 because T1 covering maximum number statements. Then whatever statements covered by T1 mark ‘0 ‘otherwise no changes.

Test case	Statements covered
T1	0,0
T2	1
T3	5
T4	0

Table 10

Weight mapping: 0,1,1,0

Now greedy algorithm selects the test case T2 because T2 covering maximum number statements and comes before T3. Then whatever statements covered by T2 then mark ‘0’ otherwise no changes.

Test case	Statements covered
T1	0,0
T2	0
T3	5
T4	0

Table 11

Weight mapping: 0,0,1,0. Now greedy algorithm selects the test case T3 because T3 covering maximum number statements. Then whatever statements covered by T2 then mark ‘0’ otherwise no changes.

Test case	Statements covered
T1	0,0
T2	0
T3	0
T4	0

Table 12

Now all the statements are covered by selected test cases T1, T2, T3. Here we can create two different ordering of the test cases because T1 and T2 occurring with same priority. main drawback of this method is not handling test case tie.

Order 1: T1, T2, T3, T4

Order 2: T1, T3, T2, T4

2.2 Genetic Approach

Genetic algorithm uses coverage information as chromosome in initial population and other effective parameter has considered for genetic loop. The following parameter considered in the genetic to optimizing test suite.

I. Genetic loop configuration

Genetic steps	Parameter
Initial population	Coverage information .if test case T_i covering statement S_i Then chromosome is '1' otherwise '0'
Fitness values	For test case $t_i(1,2 \dots n) = \sum S_i * \text{weight}$
Selection	Tournament selection
Crossover	Ist and II nd bit form first testcase IV th and V th bit from second test case
Mutation	First and third bit of crossover result
Termination	Target number of test cases found (in this case study target =3)

Table 14

II. Case Study

	STATEMENT	WEIGHT
Genetic example(a,b,c){		
If (a<b) {	S1	0.2
If(a==c){	S2	0.2
Print("crescent")	-	
}}		
Else If(b>c){	S3	0.8
If(b==a){	S4	0.8
Print("university");}		
Else{		
c=15;	S5	0.1
Print("chennai");	-	
}}		

Program 4

III. Initial test case and reduced test case

(i) Initial population

Test case	S1	S2	S3	S4	S5
T1	0	0	1	1	0
T2	1	0	0	0	0
T3	0	0	0	0	1
T4	0	0	1	0	0

Table 15

(ii) Fitness function

T1	T2	T3	T4
1.6	0.2	0.1	1

Table 16

(iii) Selection

Here selection uses tournament selection through roulette wheel. Initially there are two test cases selected from this wheel. But after the first iteration, it selects only one test case and that combined with parent test case to get optimized one. Here assume that T1 and T3 are selected in the first iteration.

T1	0	0	1	1	0
T3	0	0	0	0	1
T1 OR T3	0	0	1	1	1

Table 17

After combining T1 and T3, statement 1 and statement 2 are not covered, so we need crossover operation.

(iv) Crossover

T1	0	1	1	1	0
T3	0	0	0	0	0
T1 OR T3	0	1	1	1	0

Table 18

After cross over operation, some of the statements are not combined by combining T1 and T3 that shown in table 18. It needs mutation process to more optimized one.

(v) Mutation

T1 OR T3	1	1	1	1	0
----------	---	---	---	---	---

Table 19

Still last statement not yet covered .For the further iteration ,select T1 from this iteration because T1 has highest fitness function and select one more form roulette wheel for optimization .This process is continue until target reached or fixed number of iteration completed. In this case study, after completion few iteration, GA returns T1, T2, T3, T4 as prioritized test cases test cases.

3. Analysis of Test Prioritization Techniques

Technique	Advantage	Disadvantage
Test Case Prioritization using Extended Diagraphs	It uses Graphical User Interface (GUI) based applications and best the outcomes and best in class test case prioritization approaches.	1.it is not suitable to test the procedure oriented language 2.not handling test case tie
A similarity-based approach for test case prioritization	It based on previous used test cases. It is effective one for both procedure and object oriented language	1. Not handling test case tie
Greedy	Loop and control statements are tested easily	1. It need more time to prioritize the test cases. 2. not handling test case tie
Genetic	1. More effective optimized test case 2. coverage ratio is more	Sometimes lack of parameters has considered for genetic loop
Jtop	Fault detection rate is improved	Difficult to make cluster for loop testing
orthogonal array approach	Coverage ratio is more	Some effective test cases are not properly arranged
G-Rank Test	based on coverage of complex code 1.More complex part code are tested effectively	1.Lower complex code always tested at the end of testing process 2. not handling test case tie

4. Conclusion and Future work

Thousands of techniques has been proposed for test case prioritization in regression testing but still there is some research in the prioritization techniques. In this paper , the basic idea about different types of test reduction techniques are discussed .Finally detailed overview of greedy and genetic approach for test case prioritization are discussed ,presented and performance analyzed. From these analyses, research concluded that result of genetic approach is more

optimal than other conventional approach but still there is some issue in the genetic approach due lack of parameter used in the genetic process .In the most of conventional techniques, test case tie handled by random approach but this not suitable when we have more number test cases occurring with same priority. So in future this research going to focus this problem.

References

1. "Amit Srivastava, Er. Rajesh Tripathi", Test Case Reduction For Regression Testing,IJSTR,Issue 10, Oct-2015 ISSN 2277-8616,pp:75-99.
2. "Ahlam Shakeel Ahmed Ansari¹, Prof. K. K. Devadkar², Dr. Prachi Gharpure³ An optimized technique for test suite refinement in regression test, International Journal of Advancements in Research & Technology, Volume 2, Issue 7, July-2013,pp 263-267.
3. "Ali Yamuc", Solving Test Suite Reduction Problem Using Greedy and Genetic Algorithms, ECAI 2017 - International Conference – 9th Edition,Electronics, Computers and Artificial Intelligence, 29 June -01 July, 2017.
4. "Vaibhav Chaurasia¹, Yogita Chauhan²", Thirunavukkarasu K", A Survey on Test Case Reduction Techniques, International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064,pp-584-589.
5. "Dr. R. P. Mahapatra¹, M. Mohan² and A. Kulothungan³", "Effective Tool for Test Case Execution Time Reduction", 2009 International Symposium on Computing, Communication, and Control (ISCCC 2009) Proc .of CSIT vol.1 (2011) © (2011) IACSIT Press, Singapore
6. Rothermel, G.; Harrold, M.J., "A safe, efficient algorithm for regression testselection," Software Maintenance ,1993. CSM-93, Proceedings., Conferenceon , vol., no., pp.358,367, 27-30 Sep 1993
7. Leung, H.K.N.; White, L., "Insights into regression testing [software testing],"Software Maintenance, 1989., Proceedings., Conference on , vol., no.,pp.60,69, 16-19 Oct 1989.
8. A test-suite reduction approach to improving fault-localization effectivenessGong DandanWangTiantian Su Xiaohong Ma Peijun School of Computer Scienceand Technology, Harbin Institute of Technology, Harbin 150001, ChinaComputer Languages Systems Structures (Impact Factor: 0.3). 01/2013;39(3):95108. DOI: 10.1016/j.cl.2013.04.001
9. Mohapatra, S.K.; Prasad, S., "Minimizing test cases to reduce the cost of regressiontesting," Computing for Sustainable Global Development (INDIACom),2014 International Conference on , vol., no., pp.505,509, 5-7 March2014.
10. "Marwah Alian, Dima Suleiman, Adnan Shaout", Test Case Reduction Techniques – Survey, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 5, 2016,pp:264-275.
11. "Shrutakeerti Behura and Ajit Kumar Nayak", **Advanced Engineering Informatics**, Elixir Adv. Engg. Info. 40 (2011) 5183-5185,pp:5183-5185.

12. “Mrs.B.Subashini,, Dr.D.JeyaMala”, Reduction of Test Cases Using Clustering Technique,IJIRSET, Volume 3, Special Issue 3, March 2014, ISSN (Online) : 2319 – 8753,pp:1992-11996.

