An Efficient Software Testing By Test Case Reduction, Prioritization And Parallelization

Mr.Pradeep Udupa[1], Dr.S. Nithyanandam[2]
[1, 2]Prist University Thanjavur Vallam

Abstract- *Software Testing is the process of verifying and validating system with goal of detecting and eliminating errors or it involves validating an attribute and to see it generate expected output and required outputs. here apfd,prioritization technique, test case rank, test case reduction used and algorithm is developed to optimize the testing efficiency &reduce the execution time by reducing no of test cases, prioritization, fault detection, and further prioritized parallelization concept is used to maximize productivity.*

Keywords- Test Optimization, Prioritization, Optimization.

I. INTRODUCTION

1.1 TESTING OVERVIEW

SOFTWARE Testing is the technique of detecting, exploring then correcting errors. It is used to ensure software quality and completeness. Here the goal is:

Minimize total test runs. As no of test case increases it takes more time to test, we try to minimize test cases then prioritize and optimize.

1.2. PROBLEM DESCRIPTION

A. ISSUE OF INTEREST

As the size of program increases and no of instructions increases time and expenses also increases so derived test case should detect and explore maximum faults, hence developing technique to optimize performance is big challenge.

B.PROBLEMS INTERESTED

1. Diminish no of possible test cases
   This reduces effort, complexity, and validity
2. Prioritize test cases to increase speed
3. Calculate apfd to improve performance & quality
4. Diminish no of runs
   By reducing no of possible test cases by proposed Algorithm
5. Diminish total and average time required to execute by adopting parallelizing techniques

Steps :=>

1) Take a code segment
2) Generate flow graph
3) Compute cyclometic complexity
4) Find independent paths
5) Use algorithm to reduce test cases
6) Generate test cases for each variable
7) Analyze all independent paths
8) Prioritize test cases
9) Parallelize to reduce time

1.3 MOTIVATION

Testing is a major phase in developing software product. After completing a software product, software developers invest more effort in testing it. It includes designing test case plan, producing test inputs for exploring program behaviors According to a 2002 NIST report, it is estimated that over $22 billion of the costs of software errors could be removed by incorporating better software testing methods. Current inefficient testing methods often still take up half or more of a software project's budget. Out of all those testing costs, generating test inputs for running a program takes a huge amount of time. Those include, but are not limited to, generating inputs for exploring every possible program behavior, valid inputs, invalid inputs, and generating performance testing data. To generate these data, software firms need to hire professionals who know to produce inputs which are very expensive.3) Simultaneously running and early exploitation of errors.

Given a large number of existing test cases, our proposed method reduce and rank them such thatTest cases can explore more no of faults and program behaviors in a given time, and reduce overall execution time.

1.4 CONTRIBUTIONS

The main goal of this dissertation project is to investigate how we can improve the Efficiency of test case execution. It is done by

1) Diminish no of test case
2) Ordering test cases based on priority.

1.5 ORGANIZATION

This dissertation is organized as follows. Chapter2 introduces some software testing concepts and techniques used in the rest of dissertation and summarize some related work. Chapter3 describes our methodology and experimental results for test case reduction. Chapter4 presents test case prioritization and selection methods and experimental results for them. I conclude this dissertation in Chapter5.

II. BAGROUND STUDY

2.1 CHECK LISTS OR TEST CASE

A test case is designed to test whether system works properly or not basically it is a one step, or it is a no of steps, it is used to test the correct behavior/operations and features of an application. An expected result or expected outcome.

2.2 VALIDATION SUITE

A validation suite is a group of test cases used to validate system with respect to given constraints or requirements it used to validate whether system gives expected output.

2.3 TESTING CONTROL FLOW

Used to test every possible path, It can be used when no of paths are more and testing more no of paths are complex and time consuming.

2.4 INDEPENDENT PATHS

It is unique path in the program which is used to test specific and different criteria and generate test cases.

2.5 CYCLOMATIC COMPLEXITY

It is used to derive no of independent path existing in program or unique path in control flow graph
Cyclomatic complexity =no of Edges-nodes+2

2.6. WHY TO LESSEN OVERALL TEST SUITS?

1. Bigger the test cases more the complexity
2. Large the test cases more probable no of errors
3. Error tracing is to be performed
4. Huge no of testers are needed
5. It will take long time

2.7. NEED & SCOPE OF THE STUDY

1) To explore maximum no of errors.
2) To prioritize, reduce test cases, and run time.

2.8. WHY TO USE PROPOSED TECHNIQUE

Existing techniques such as DDR, basis path test leads to more number of test cases and inefficient so we

1. Diminish no of test cases
   Proposed technique reduces overall test cases, effort of testing, executing and validating test.
2. Diminish total no of test runs
3. Decrease time required to run test cases

In proposed research we try to reduce no of test cases by finding (how)

Min, max, and constant values in the entire test cases though finding no test paths.

Here an efficient algorithm is written to reduce total no of test cases an analysis is made with other algorithm to prove proposed algorithm has greater efficiency and takes less no of test cases to execute and time required to execute and cost required to execute will be less.

Here first we design a flow graph for algorithm then find all independent path in program next for each independent path we design range of test cases and no of test cases.

2.9. PROPOSED ALGORITHM (HOW)

In this we use following steps to decrease no of test cases

1) Detect criteria's from begin to end nodes. A criteria can be (>, >=, <, <=, ==! =)
2) Detect the variables with high and low Values in the path, then the large variable is given high value and small variable is given low value.
3) Detect fixed values in the path, and allot to variables found in path, and then use parallelization technique to run test cases parallel.
4) Then employ obtained range to derive all test cases.Next we reduce test case by above algorithm

then prioritize test case by giving rankings using test case ranking

III. LITERATURE REVIEW

In last few years there were many publications which discussed the concept of test case reduction. In this section different test case reduction methods and related works are discussed.

(A) Constraint-based test data generation by Richard A. DeMillo and A. Jefferson Offutt presented an approach to test data generation that uses control- flow analysis, symbolic evaluation. it uses control- flow analysis,

Table 1. Contrast among different depletion approach for test cases.

S.NO	*TEST REDUCTION TECHNIQUE NAME*	*ADVANTAGE*	*DISADVANTAGE*
1	CONSTRAINT BASED TESTING	it uses control- flow analysis, symbolic evaluation and reduces no of test cases based on criteria	1.More no of test cases 2. Time consuming 3. More expensive 4. Less efficient 5. More effort 6. No parallelization
2	*D.D.R*	Achieved more depletion in test cases	1. less test cases compare to constraint bases 2. comparatively Less Time consuming 3.less expensive 4.less effort 5.no parallelization
3	*PING PONG TECHNIQUE*	Assure domain Coverage and cost effective	Time consuming, expensive technique more memory ,time and efficiency is required in executing the test cases
4	*Prioritize techniques no order, reverse order*	Attempt to detect possible no of errors and contribute in performance	Less efficient compare to proposed technique in terms of performance

Symbolic evaluation and reduces no of test cases based on criteria.

(B) dynamic domain reduction (ddr) by a.jefferson Offutt zhenyi jin uses get split algorithm which divides domain to reduce overall domain range and reduce test runs and total time requred. Achieved more depletion in test cases but it is less efficient and time consuming and more effort needed.

(C).PING-PONG TECHNIQUE

This technique selects less minimum no of test cases by arranging differently, works using heuristic method which wont ensure best result but give good result in given time, by contrasting the set of values of goal state and set of states of achieved values and assure domain coverage. But its time consuming and more expensive technique

(D) TEST CASE REDUCTION USING PARALLELIZATION

Many techniques are represented previously in literature but in our technique we used test case reduction approach, prioritization, fault detection and parallelization where min, max, constant variable in all path are found and later more than one test case are made to run in parallel fashion which has greatest percentage of reduction in terms of no test cases and execution time required for running parallelization, debugging.

IV. EXISTING METHODOLOGY
DDR TECHNIQUE

A step follows assume given domain is i1(0..30),j1(0..50),k1(0..40)

1. detecting all criteria's from begin to end.
2. Evaluate split point value for range of domain and for all variable satisfying criteria.

Then as per split vale we segregate into two intervals. Ma1=0 to 15 and 16 to 30 ma2 into 10 to 30 and 31 to 50 &final interval by using splitting is ma1 0 to 10 and 11 to 30 ma2 31 to 50 ma3 is 10
So total test cases=31*1+31*20=651

V. PROPOSED METHODOLOGY

We first reduce test case by our given algorithm Then prioritize test case by giving rankings for Test cases then find APFD which will prove that Ours technique over performed Then existing Techniques, then parallelize our test cases to Reduce time and cost involved. Here first we find No test paths then from each path we find min,Max, and constant

values and derive our Reduced test cases by using Steps given below Then further execute them parallel.

Table 2.Derived range of values for variable I1, J1, K1.

TEST TABLE			
VARIABLE I 1	*VARIABLE J1*	*VARIABLE K1*	*TEST CASES /PATH*
0 to 30	31 to 50	10	T1 /P1
0 to 9	10 to 50	10	T2/P2
10 to 30	0 to 30	20	T3/P3
30	0 to 50	20	T4/P4

VI. RESULT EVALUATION

Here Proposed Techniques is contrasted with the Existing Technique Get Split with respect to

1) generated total checklists or test cases
2) Total depletion in test cases
3) Comprehensive bugging time
4) Fault detection rate

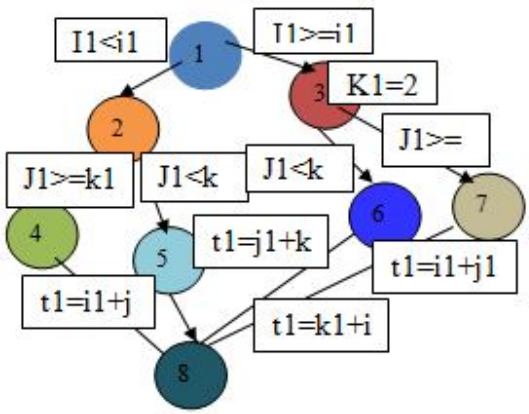


Figure 1.CONTROL FLOW GRAPH

Table 3. No of faults exposed & time required for each faults.

Test cases/faults	T1	T2	T3	T4	norder	revord	prop
F1				*	4	1	1
F2				*	4	1	1
F3	*		*	*	1	4	1
F4	*	*	*	*	1	4	1
F5			*		3	2	3
F6		*			2	1	4
No of faults	2	2	3	4			
time	3	5	7	9			
severity	4	6	8	10	15	13	11

Then for faults in table 3 severity given which is mentioned in Table3.

RFT=Nj/TIMEj*10 as in Table5.
PFD=NJ/total no of faults*10
RDA=NJ*SJ/TJ
TCR=RFD+PFD+RDA as in Table4

Table 4. Evaluated test case ranking value.

Test cases	TCR=RFD+PFD+RDA
T1	8.5
T2	10.6
T3	10.33
T4	10.5

Table 5. Fault rate,PFD,RDA and test case rank.

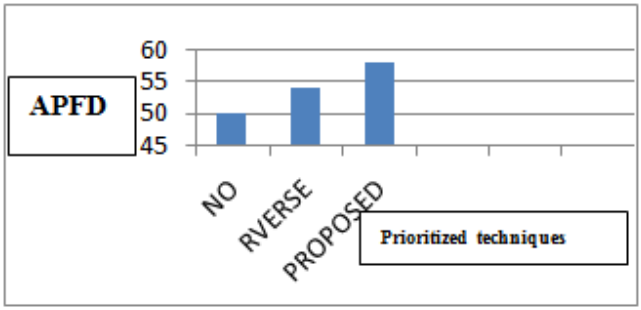
Test cases	RFT	PFD	RDA	TCR
T1	6.66	1.81	2.66	11.13
T2	4.0	1.81	2.4	8.21
T3	4.28	2.72	3.42	10.42
T4	4.44	3.63	4.44	12.51

Table 6.Test case execution order of different ranking Approaches.

NO ORDER	REVERCE ORDER	PROSED ORDER
T1	T4	T4
T2	T3	T1
T3	T2	T3
T4	T1	T2

APFD=1-(TF1+TF2+…TFM)/M*N+1/2*N
For no order apfd=1-(4+4+1+1+3+2)/6*4+1/2*4=1-.625+.125=50% For reverse order apfd=1-(1+1+4+4+2+1)/6*4+1/2*4=1-.3125+.125=54%
For proposed order apfd=1-(1+1+1+1+3+4)/6*4+1/2*4=1-.4583+.125=58% 2 as in

Table 7.Calculated apfd values of different tRanking approaches.



NEXT we parallelize test cases by referring Test table given above Now variable I used in 3 paths so range =total no of interval/3 Now variable j used in 4 paths so range =total no of interval/4 Since c is constant we not divide c So range of I spitted into 3 parts i1) 0…..10 i2)11…..20 i3)21….30 similarly range of j spitted into 4 parts j1) 10…..20 j2)21…..30 j3)31….40 j4)41…50 Now when we execute them parallel fashion we have Total no test cases=[31*51*41]*4=259284.

Reduced test cases=[31*20*1+10*41*1+21*31*1+1*51*1] =[620+410+651+51]=1732,Assume each test case take .5 second then Without parallelization execution time is 1732*.5=866 So total no of test cases 259284, execution time129642.Reduced test case for sequential execution no of test case1732, execution time 866 But with PRIORITIZED parallel execution test cases=1732 and execution time=433 and fault detection rate is more, and time required to detect THE FAULTS WILL BE considerably low because test cases are exposed in prioritized order as mentioned in Table6.

VII. CONCLUSION

Each algorithm has its own significance as well as drawbacks ddr works on specific domain & split points, Ping pong and other existing techniques results in more no of test cases, compilation, time, effort, cost but proposed technique over performed by reducing no of test cases, prioritizing, revealing more faults by assigng test case rankings, apfd calculation and finally prioritized reduced test cases as in Figure 2,and by giving test case rankings to achieve optimized performance and parallelized and prioritized test cases to reduce overall running time and cost.

VIII. LIMITATION

This executes serially and each and every path is to be examined so it will take more time and memory to store the result, it is effective when the variables are there with constant and fixed values it works well for parallel execution where we can save memory and increase speed of execution.

REFERENCES

[1] R. Wang, B. Qu, Y. Lu.(2015).“Empirical study of the effects of different profiles on regression test case reduction”. IET Softw, 9(2), pp.29–38.

[2] B. Boehm and L. Huang.(2003).“Value -Based Software Engineering: A Case Study” .IEEE Computer, 36(1), 2003, pp.33-41.

[3] Arnicane, V.(2009).“Complexity of Equivalence Class and Boundary Value Testing Methods”. International Journal of Computer Science and Information Technology,751(3), pp.80-101.

[4] Abhijit, A., Sawant1, P. H. Bari2 & P. M. Chawan3.(2012).” Software Testing Techniques and Strategies”.IJER 2(3), pp. 980-986.

[5] Jeng B., Weyuker E. J..(1994).“A Simplified Domain - Testing Strategy”. ACM Trans. Softw. Eng. Methodol.3(3), pp.254–270.

[6] T.Gyim´othy, A. Besz´edes, and I. Forg´acs.(1999). “An efficient relevant slicing´ method for debugging”.SIGSOFT Softw. Eng. Notes, 24(6),pp. 303–321.

[7] S. Biswas and R. Mall.(2011).”Regression test selection techniques: A survey”. Informatica 35(2),pp. 289–321.

AUTHOR’S BIBLIOGRAPHY

MR.PRADEEP UDUPA completed his MTECH (CSE), MCA MPHIL & PERSUING PHD (CSE) his field of interest includes software engineering, testing, operating system, simulation, dbms, wireless communication, adbms,
Oops currently working as assistant professor in mes eng Kerala.