Study of Optimization and Prioritization of Paths in Basis Path Testing

Gaurav Siwach
Research Scholar
M.Tech(CSE)
Amity University, Haryana

Sunil Sikka,PhD
Assistant Professor
Deptt. of CSE
Amity University, Haryana

Priyanka Makkar
Assistant Professor
Deptt. of CSE
Amity University, Haryana

ABSTRACT
Testing has become an essence part of the software development life cycle. Structural testing is a testing type, which focuses on the control flow of the program. Basis path testing is a kind of structural testing which derives a set of basis paths from control flow graph. These basis paths ensure that every statement of the program under test has been executed at least once. This paper studies the different techniques used by different researchers for the prioritization of these paths. The optimization and prioritization of the paths increases the probability of finding more errors within the limited resources.

Keywords
Basis path testing, Control Flow Graph, Cyclometic Complexity

1. INTRODUCTION
Software testing is a necessary and integral part of the software quality process. It is estimated that 80% of software development cost is spent on detecting and fixing defects [1]. It is a necessary evil in the process of software development. Moreover, it adds nothing to the functionality of the software. Even with intense testing, we cannot give assurance that our software is 100% free from errors. Software testing is done to enhance the quality of the software and this is achieved by detecting and fixing the bugs. There are many methods of testing which are mainly grouped in Black box testing and White box testing. Black box testing also called behavioral testing focuses on the functionality of the software. It uses some input condition which will fully exercise the functionality of program. It uncovers the errors of following categories:(1) incorrect or missing functions, (2) interface errors, (3) errors in data structures or external database access, (4) behavior or performance errors and (5) initialization and termination errors[2]. On the other hand White Box Testing also called glass box testing focuses on the structure of the software. That is why it is also called structural testing. Test cases are designed to check the coverage of each path, branch or statement [2]. Basis path testing is also a white box technique developed by Thomas MaCabe. This method generates a set of linearly independent paths, which are called basis paths. These paths can be derived from Control Flow Graph (CFG). Basis Path Testing is a more rigorous software testing criterion typically used for program unit testing. To uphold the thoroughness of testing, the tester has to design the test cases, but sometime test cases grow in much large size that they drain all the testing resources. In most cases, the tester has to determine test cases manually. This becomes more difficult when the program under test has complex branching structure. So basis path testing helps in tackling above both the problems. The method devised by McCabe to carry out basis path testing has following four steps.

Step 1. Generate CFG of program under test.
Step 2. Compute the cyclomatic complexity from CFG.
Step 3. Generate the basis set of paths which are equal to cyclometic complexity.
Step 4. Design test cases for each of the generated paths.

CFG is the graphical representation of the control structure of the program under testing. It consists of a set of nodes N and a set of edges E. Each node represents a set of procedural statement and is denoted by a circle labeled with a name or number. Edges in CFG represent the control flow within the program. It is denoted by an arrow and must terminate at a node. Any executable path in module's CFG would start from the entry node and end at the exit node. In a CFG, a node including condition is called a predicate node, and edges from the predicate node must converge at a certain node. For example Fig 1 shows the CFG of following pseudo code to display roots a quadratic equation of the form $ax^2+bx+c=0$.

```
1. Input(a,b,c)

2. D=b^2-4ac

3. if(D>0) then

4.        r1=( -b+ sqrt(D) ) / (2*a)

          r2=( -b-sqrt(D) ) / (2*a)

          print r1.r2

5. else if(D=0),then

6.        r1=(-b) / (2*a);

          r2=r1

          print r1,r2

7. else print "Roots are Imaginary "

8. end
```

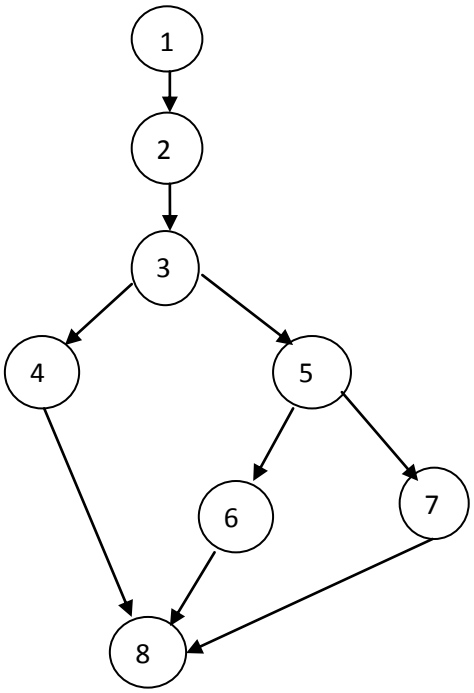

Fig 1: Control Flow Graph for solving quadratic equation

The cyclomatic complexity also known as structural complexity calculates the number of independent paths through a program. It provides the upper bound of the number of test cases that must be designed, in order to ensure that all statements have been executed at least once and all conditions have been tested. Generally cyclomatic complexity of a graph G is denoted by V(G).

There are three methods to calculate the Cyclomatic complexity. One of these is V(G) = e – n + 2, where e represents the number of edges and n denotes the no. of nodes in the CFG. According to second method V(G) = P + 1 where P is no. of predicate nodes in CFG. According to the third method V(G) is equal to number of regions in the CFG. The outside area of CFG is also a region.

The rest of the paper is organized in two sections. Section 2 presents the review of various research works related to basis path testing and section 3 presents the conclusion of the paper.

2. LITERATURE REVIEW

A lot of researchers have given various techniques for finding the feasible paths and prioritization of paths. Prioritization of execution of paths increases the chances of finding more errors. It also helps in selection of more appropriate test paths if there is less time for testing. Optimization of paths helps in removing the infeasible paths from basis set of paths. Infeasible paths defined as the path that cannot be executed by any set of possible input values. Infeasible paths mainly crop up when decision nodes are in series connection and variables involved in decision nodes before and after have a certain degree of dependency [3]. This section reviews some of the techniques for prioritization of paths.

Z.Guangmei et al. [4] deduced a method of generating the basis set of path by applying Depth First Search (DFS) on the CFG of the program . The given procedure first visits that path whose tail node is the start node and the head node of this edge is stored on the top of stack (A stack is used for storing the head nodes of the visited edges). Next it visits the subsequent unvisited edge of current edge and store their head node onto the stack. While developing a basis path, when a multi-in-degree node encounters and there is no sub-path from this node to the exit node then that path (from start node to multi-in-degree node) is recorded. And when a sub-path is found to exit node, a new basis path is generated by merging the recorded sub-path and current sub-path. Hence, the given method readily generates the basis set of paths.

T. Lertphumpanya et al. [3] proposed a method which applies basis path testing on Web Service Business Process Execution Language (WS-BPEL) services. WS-BPEL is an OASIS standard executable language for specifying action within business processes with web services. The method used here generates the test suit for basis path testing of WS-BPEL. This test suite includes test cases, auxiliary state services that assist in test and stubs of the constituent Web Services within the flow. A tool is also specified in the paper which will be used by service tester.

P.R Srivstava et al. [5] used the coordinated behaviour of real ants and applied to some artificial agent which collaborate to solve the complex computational problems. The proposed algorithm selects all the optimal paths in between entry node and exit node. And this selection is based on the probability of each path. Higher is the probability higher the chances of selection of path. The probability of each path depends on the feasibility of path, Pheromone Value, and heuristic information of the path.

S.Bardin et al. [6] focused on a very important problem called path explosion phenomenon. This is a common problem in path based testing with item coverage. It uses three heuristics to tackle this problem which are both easy to implement and cheap enough to execute. Moreover each heuristic deals with a different source of path explosion. These three heuristic are:

- The **Look Ahead (LA)** heuristic: when it find out that from current control location, there is no possible path to some uncovered pat, it stops the current path exploration.
- The **MaxCall Depth (MCD)** heuristic: It tackles the path explosion problem which occurs due to nested calls.
- The **Solve First (SF)** heuristic: It gives priority to solve shorter path prefixes.

Y. Chen et al. [7] used Genetic Algorithm for automatic test data generation. Use of the appropriate fitness function is very important for any genetic algorithm as it increases the chances of finding a solution and possibly uses few system resources. This paper compared such two fitness function namely BDBFF and SIMILARITY. The applied Genetic Algorithm based approach for automatic test data generation using above two fitness function, is applied on triangle classification problem. The experimental results showed that BDBFF based approach achieved the target path within less test data in comparison to SIMILARITY based approach.

D. Jeya Mala et al. [8] proposed a method called Artificial Bee Colony(ABC) optimization which is a non-pheromone based intelligent optimization technique for test suits optimization. The model uses three agents: Search Agent, Optimizer Agent & Selector Agent in correspondence to three group of bees are Employed, onlooker & scouts. The paper used path coverage as the test adequacy criterion for improving the quality of the test cases. ABC based approach also removed the problems which were faced in Ant Colony Optimization like continuous pheromone update, computational time and memory overheads.

.Z. Zhonglin, et al. [9] introduced a approach for avoiding the infeasible paths in the basis set. It gave the reason of occurrence of infeasible path is that when two decision nodes are connected in series then there are more chances of occurrence of infeasible paths. It modified the CFG by replacing the edges which are data dependent with the dotted edge and control edges with solid edge. Then it chose the shortest path as a baseline path having more predicates node and applied the flip operation to extract feasible paths.

D. Gong et al. [10] proposed a very efficient method for detecting the infeasible paths. The method first scans the whole program for conditional statements and for the correlation between them. Then it applies a theorem for detecting33 the infeasible paths. A case study was also used for the verification of results. However the experimented results are preliminary as the method was performed on simple program with simple structure.

Qingfeng, et al. [11] proposed a method which deals with the infeasible paths in basis set. This method modified the generated CFG by connecting the causal paths of two series judgment parts and skipping the intermediate nodes. The reason behind the occurrence of infeasible path is that the two decision nodes are in series connection.

Minjie Yi [12] proposed a method for automatic generation of test data by mixing two most important techniques: Ant Colony Optimization and Genetic Algorithm. Author proved that this method produces more efficient test data in terms of validity and quality.

Madumita Panda et al. [13] found out the number of feasible paths present in the CFG using cyclometic complexity. And it compared actual no of paths covered by test cases which were evolved using three meta-heuristic search algorithm Genetic Algorithm, Artificial Bee Colony Optimization algorithm and Differential evaluation. Authors also showed the effectiveness of all three search algorithms. Study demonstrated that overall Differential Evaluation is more effective in comparison to other two techniques. In terms of time complexity the execution time of Differential evaluation is better than other two. But the search space exploration of paths is slightly better for Artificial Bee Colony Optimization than Differential evaluation.

S. Srivastva et al. [14] proposed a model for prioritization of basis paths in basis path testing using Ant Colony Optimization (ACO) algorithm. It used the CFG to represent the software under test. After execution of this algorithm it would first calculate the probability of each path then give the highest priority to that path which would have highest probability.

Y.Suresh et al. [15] developed a system which automatically generates the test data using the soft computing technique called Genetic Algorithm (GA). The proposed system first calculates the basis paths from the CFG and then produces the most favorable test data from these basis paths automatically. Initially, the GA generates the population randomly (Here population is considered for test data). Then it calculates the fitness value of each individual chromosome using some fitness function. On the basis of that value it performs mutation and cross-over (GA operation) to enhance their fitness value. These enhanced chromosomes makes the next generation. This process continues until all individual chromosome reach to their maximum fitness. So this automatic generation of test data reduces the time, cost and effort of the tester.

Himanshi' et al. [16] elaborated the work of Srivastva giving a varying Ant Colony Optimization Algorithm. In comparison to work of Srivastva this method gave priority on the basis of probability of path as well as path length.

Y.D.Salman et al. [1] used UML state chart diagram for design specification of the software under test(SUT) and thereafter generating the test cases. It doesn't generate the test cases directly from UML state chart diagram. First, it converts the state chart diagram of SUT to an intermediate Testing Flow Graph(TFG) , and then from TFG it generates test cases. The paper first found out the possible test paths from TFG and reduced them into feasible test paths.

Ghiduk et al. [17] proposed a different technique for automatic test data generation using the Genetic Algorithm. The GA technique applies the idea of dominance relation between nodes to define a new fitness function to evaluate the generated test data. The technique has completed many objectives in the sense that it is effective in achieving coverage of test requirement and also in reducing the size of test suits.

G. Balakrishnan et al. [18] applied the method of abstract interpretation to find the infeasible paths on the basis of their semantics. It deduced the feasible paths in the CFG by using path insensitive forward and backward run sequence.

2.1 Analysis of Different Techniques Discussed

In this section analysis of different techniques discussed in this paper is provided. This analysis is done in terms of problems countered and the methodology used to solve that problem. This is provided in tabular form in table 1.

Table 1. Analysis of different techniques

Authors	Problem Countered	Methodology Used
Z.Guangmei et al.[4]	1. Automatic generation of basis set of paths 2. To prove that path generated are basis paths	Depth First Search of CFG
P.R Srivstava et al.[5]	1. Automatic selection of optimal path 2. Path prioritization	Ant Colony Optimization(ACO)
S.Bardin et al.[6]	1. Path explosion in path testing due to loops, nested loops, conditions	1. Look-Ahead heuristic(LA) 2. Max-Call Depth heuristic(MCD) 3. Solve First(SF)
D.Jeya Mala et al.[8]	1. To increase the efficiency of test cases 2. Optimization of test cases 3. 100% path coverage with least test cases	Non-pheromone based Artificial Bee Colony Optimization(ABC)
Z. Zhonglin et al.[9]	1. Detection of infeasible path in basis path testing	Baseline method is combined with data dependency

	2. Removal of infeasible paths occurred due to data dependency	
Minjie Yi[12]	1. Automatic generation of test data for test paths	Combination of 1) Ant Colony System algorithm which is a modified version of ACO and 2) Genetic Algoritm
Himanshi et al.[16]	1. Sequencing the execution of test path among a set of test path 2. prioritizing the test path	Extension of Ant Colony Optimization which give highest priority to shortest feasible path

3. CONCLUSION

Many researchers have proposed various techniques for optimization and prioritization of test paths. Few important techniques discussed in this paper are: Genetic algorithm, Ant Colony optimization, Artificial Bee optimization algorithm, Differential Evaluation, UML State Chart based etc. However, each technique uses a different method for solving the problem but has a common aim: Automation of generation of test suits and increasing the effectiveness of path based testing.

4. REFERENCES

[1] Y.D.Salman, N.L.Hashim (2014). "An Improved Method Of Obtaining Basic Path Testing For Test Case Based on UML State Chart ", International Symposium on Research in Innovation and Sustainability(1013-5316).

[2] R. Pressman (2009). Software Engineering: A Practitioner's Approach, 7th ed., McGraw Hill.

[3] T. Lertphumpanya T. S. (2008). "Basis Path Test Suite and Testing Process for WS-BPEL", WSEAS TRANSACTIONS on COMPUTERS(1109-2750), Volume-7, Issue -5.

[4] Z.Guangmei, C. R., L.X. (2005). "The Automatic Generation of Basis Set of Path for Path Testing", Asian Test Symposium (1081-7735),IEEE.

[5] P.R Srivstava, K.M. Baby, and G Raghuram (2009). " An Approach of Optimal Path Generation using Ant Colony Optimization ", IEEE(1-6).

[6] S.Bardin, P. H. (2009). "Pruning the Search Space in Path-based Test Generation", International Conference on Software Testing Verification and Validation.

[7] Y. Chan, Y. Z. and T. S. (2009). "Comparison of Two Fitness Functions for GA-based Path-Oriented Test Data Generation", Fifth International Conference on Natural Computation.

[8] D.Jeya Mala, M. K. and R. S. (2009). " Intelligent Tester – Software Test Sequence Optimization Using Graph Based Intelligent Search Agent ", International Conference on Computational Intelligence and Multimedia Applications.

[9] Zhang Zhonglin, Mei Lingxia (2010). "An Improved Method of Acquiring Basis Path for software testing", ICCSE'10 (pp. 1891-1894).

[10] D. Gong, X. Yao (2010). "Automatic detection of infeasible paths in software testing ", IET Software(361–370), Volume-4, Issue- 5.

[11] Du Qingfeng, Dong Xiao (2009). "An improved algorithm for basis path testing". IEEE (pp. 175-178).

[12] Minjie Yi (2012). "The Research of path-oriented test data generation based on a mixed ant colony system algorithm and genetic algorithm ", IEEE.

[13] Madumita, Partha Pratim (2013). " Performance Analysis Of Test Data Generation For Path Coverage Based Testing Using Three Meta-heuristic Algorithms", International Journal of Computer Science and Informatics(2231 –5292), Volume-3, Issue-2.

[14] Saurabh Srivastva, Sarvesh Kumar, Ajeet Kumar Verma (2013). "Optimal Path sequencing in Basis Path Testing", IJACEN(2320-2106), Volume – 1, Issue – 1.

[15] Y.Suresh, S.K.Rath(2013). " A Genetic Algorithm based Approach for Test Data Generation in Basis Path Testing", The International Journal of Soft Computing and Software Engineering (2251-7545), Volume-3, Issue No.-3.

[16] Himanshi, Nitin Umesh, and Sourabh Srivastva (2013). "Path Prioritization using Meta-Heuristic Approach", International Journal of Computer Applications (0975 – 8887) Volume -77, Issue No.-11.

[17] Ghiduk, Harrold andGirgis (2007)."Using Genetic Algorithms to Aid Test-Data Generation for Data-Flow Coverage", Universal Journal of Computer Science and Engineering Technology (64-72).

[18] G.Balakrishnan, S.S (2008). "Path-Sensitive Analysis through Infeasible-PathDetectionandSyntactic Language Refinement", springer verlag(1-16).

[19] Sommerville, i.(2009). software engineering. london: pearson edition.