Model-Based Automatic Optimal Test Path Generation via Search Optimization Techniques: A Critical Review

Aye Aye Kyaw, Myat Myat Min
University of Computer Studies, Mandalay
ayeayekyaw2009@gmail.com,myatiimin@gmail.com

Abstract

Software testing plays a main role in developing software that is free from bugs and defects. Although manual tests may find many defects in a software application, it is a cost and time consuming process. If the testing process could be automated, the cost of developing software could be reduced obviously. Most of researchers generate all possible test paths. The best test path is got more beneficial than the other generated test paths because it can reduce time and cost more than others. In recent trend Model-Based test case attracts many researchers by using some search optimization techniques to produce an automated optimal test path. A review of Model-Based optimal test path generation via search optimization techniques concepts is found in the current literature. The review paper identifies the problems in usage of certain techniques and presents areas that needed future research.

1. Introduction

Software testing is an essential part of the software development life cycle. A software project is made up of a series of phases such as: i) Requirements gathering and analysis, ii) Planning, iii) Design, iv) Development or Coding, v) Testing, and vi) Deployment and Maintenance. Software testing is an activity that should be done throughout the whole development process [1].

There are many testing strategy and among them, some are Model-Based Testing (MBT) that depends on extracting test cases from different models (requirements models, usage models, and models constructed from source code), Specification-based testing (or black-box testing) that depends on requirements models, and Program-based testing (or white-box testing) that uses source code as the underlying model [7].

In recent trend, Model-Based Testing becomes a more popular approach among many researchers. Model-Based Testing (MBT) is a type of testing strategy that depends on extracting test cases from different models (Requirement model, usage model and model constructed from source code) [7]. MBT promises (1)the detection of faults at an early stage in the development cycle and (2)reduces the maintenance effort of test scripts as the test cases human and cost effort are minimized. MBT is seen usually as one form of black-box testing. MBT is suited perfectly being used in system, acceptance, and regression testing [4]. MBT have three main key technologies:

- Notation used for the model
- The test generation Diagram
- The tools that generate supporting infrastructure for the tests [5].

In today's software industry, the software test design is mostly based on the tester's expertise, while test automation tools are limited to execution of preplanned tests only. Testing effort can be classified into three parts; they are test case generation, test execution and test evaluation [2]. The test case generation is

more difficult and more challenges than the latter two parts.

Testers aim is to test all the feasible paths in the diagram of the system. But in actual testing every path is not a problem but finding out the feasible path, the optimal test path is a major challenge. If the process of testing could be automated, significant reductions in the cost of software development can be achieved. It depends on the quality/fitness and number of test cases exercised. The solution is to choose the most important and effective test cases and removing the redundant and unnecessary ones, which in turn leads to test case optimization [6]. Model based test case generation equally challenging and also many researches involve achieving optimal set of test case.

This paper gives a view on the various search optimization techniques available to generate the test cases using activity diagrams. The rest of the paper is organized as follows. Section 2 presents the fundamental steps that are used by many researchers for generating the test path with Model-Based approach. Section 3 describes the search optimization techniques to select the best test path. Section 4 discuss about these techniques. The last section concluded the paper and expressed the future work.

2. Model Based Test Path Generation

Various authors have used the following architecture for generating the test path.

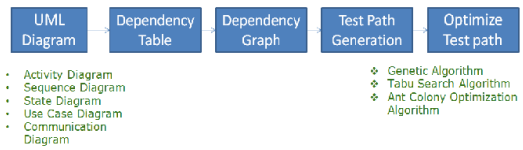

Figure 1. System Flow of Test Path Generation

Following are the steps of test case generation by using activity diagram among other UML diagram. The reasons that are used activity diagram as input are:

1. The concepts at a higher abstraction level compared to other diagrams like sequence diagrams, class diagrams, etc. are presented.
2. The results in path are the presence of loop and concurrent activities in the activity diagram.
3. It is difficult to consider all execution paths for testing [3].

The fundamental elements are actions and control nodes. Action, an individual step within an activity, can posses input and output information. The output of one action can be the input of a subsequent action within an activity. Control node is an activity node used to coordinate the flows between other nodes. Control nodes are Initial node, Flow final node, Activity final node, Decision node, Merge node, Fork node, Join node. Edges connect the individual components of activity diagrams.

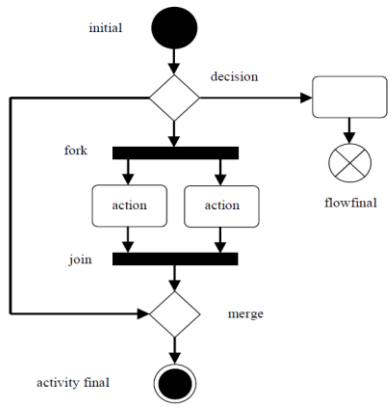

Figure 2. Elements of Activity Diagram

The description of each step in Figure 1 will be illustrated as follows:

2.1. Generation of ADT

Activity diagram is used to automatically generate the Activity Dependency Table (ADT) with all the activities. These activities include

decisions, loops and synchronization along with the entity performing the activity. ADT also consist of the input and the expected output values for each activity. Dependency of each activity on others is also shown clearly in ADT. The repeated activities in the diagram are grouped into one symbol only instead of having several symbols for the same activity.

2.2. Generation of ADG

The ADT is accomplished to automatically generate the Activity Dependency Graph (ADG). The symbols given for each activity are used to name the nodes in the ADG where each node represents an activity in the activity diagram. Since repeated activities are given the same symbol in the ADT, only one node is created for them no matter how many times they are used in the activity diagram. This will decrease the search space in the ADG. The transitions from one activity to another are represented by edges in the ADG. The presence of an edge from a node to another is determined by checking the dependency column in the ADT for the current node's symbol. Specifically, if it contains the previous node's symbol then an edge from the previous node to the current one is drawn in the ADG.

2.3. Generation of Test Path

The generated ADG applied to obtain all the possible test paths. A test path is composed of steps represented by successive symbols/nodes (representing the activities) forming a complete path from the start node in ADG to the end node separated by arrows. Details are then extracted from the ADT and added to each node in the test path to obtain all the final test cases. Each node in the test case is accompanied with its input and expected output. Besides, the whole test case will be accompanied with its initial input and final expected output [7].

3. Optimal Test Case Generation

Many researchers have been successfully proposed test case generation for various software under many circumstances, using mainly search optimization techniques such as Genetic Algorithm, Ant Colony Optimization Algorithm, Tabu Search Algorithm, etc. The following papers are based work evolved under model based test case generation.

3.1. Tabu Search Algorithm

In [2], Shantghi and Mohan Kumar proposed an approach for generating test cases for object oriented software that uses Tabu Search technique to design and derive test cases. The Tabu search technique was applied to generate optimal test case.

Their paper is described that their approach is significant due to the following reasons. First, their approach can detect errors like errors in loop, synchronization faults easier than the existing approaches. Second, test case generated in their approach may help to identify location of a fault in the implementation, thus reducing testing effort. Third, their heuristic method for test path generation inspires the developers to improve the design quality, find faults in the implementation early, and reduce software development time. Fourth, following their approach, it is possible to build an automatic tool easily.

3.2. Ant Colony Optimization Algorithm

Saurabh Srivastava, Sarvesh Kumar and Ajeet Kumar Verma proposed a technique for optimal path sequencing in basis path testing. This approach is used to generate optimal paths using control flow graph (CFG) and Cyclomatic Complexity for finding the number of feasible paths. This paper applied ant colony optimization algorithm to prevent from selecting infeasible

path and also prioritize the feasible paths i.e. which path is to be executed first to optimize time and complexity. They used an idea of prioritizing the feasible path or each and every path gained by using control flow mechanism in order to find optimal path and the algorithm will assign some value (priority) to each feasible path according to which these paths will be tested on the basis of priority. The path with higher priority will be executed first and with lower priority thereafter. With great ease of the ant colony optimization algorithm good results can be got and thus basis path testing can be improved, but there are areas where it can be extended such as if repetition of states or nodes can be controlled which have multiple paths. After execution of this algorithm, it automatically selects the best path sequence [8].

3.3. Genetic Algorithm

This section is described two proposed approaches to generate the optimal test path using genetic algorithm.

Model Based Test Case Generation Technique Using Genetic Algorithms presented by Mary Sumalatha and Raju based on UML activity diagram. Finding all paths genetic algorithm operations are applied to obtain the best test case [10]. In case study of their paper, the expected results should be accurately. As the above generated new individuals are not part of the generated paths Path 1 is the best node that is selected. By really Path 4 that is covered the whole of system should be the best path.

Shanthi and Mohan Kumar [3] proposed a new model based approach for automated test cases for UML activity diagrams using genetic algorithm. With the help of ADT test path are generated, by applying the GA most prioritized test case are generated. For each node, the number of incoming nodes and the number of outgoing nodes are calculated and then evaluate fitness values. Next, the initial test data is selected by randomly. Lastly, the Prioritized test path is generated. The generated test cases apply the branch coverage criteria and the cyclomatic complexity coverage.

4. Discussion

To sum up all, many authors have proposed methods for optimizing test path. Some have used Tabu method, some have used ACO method, and some also GA method.

In Tabu search algorithm, the greatest test path value is chosen as the best test path, their results may be changed. The test path values are based on assigned weight value. A positive effect of using Tabu method is that it can process within short time. A negative effect is that the style of assigning weight value is changed; the test path values are changed.

Genetic Algorithm simulates natural evolution by mimicking processes that the nature uses, like selection, crossover and mutation. GA operates on an initial population by applying the principle of survival of the fittest. It produces either a better or better approximation to a solution. The GA operates either cross over or mutation process based on the random number iterately until when some criteria are satisfied, or when a particular point in the search space is encountered. The obvious benefit of GA is that it can produce the more accurate result owing to iteration of crossover or mutation process until the result is satisfied by testers. One drawback is that the generated test path by using GA may be changed because of the number of crossover process or the number of mutation process. Moreover, an additional drawback of using GA is that it consumes more time than Tabu search algorithm.

Theoretical analysis of Ant Colony Optimization is difficult. ACO use sequences of random decisions and the probability distribution changes by iterations. ACO is more suitable for

dynamic problems such as network routing. However, the strength of ACO is that it can find the optimal test path. Its weakness is that it can't guarantee its iterations and time.

The comparison of the search optimization algorithms are shown in Table 1. According to the nature of ACO, the processing time of ACO will take more than the other two algorithms such as Tabu search and GA. Because ACO will perform many iterations time to find the best test path. Similarly, according to the nature of these algorithms, complexity of ACO is more complex than two others algorithms. The accuracy of results (the best test path) of ACO is better than other two, i.e, guarantee of ACO is more. This is because calculation of Tabu and GA depends on assigned weight value and the number of mutation or crossover process.

Current work will take the advantages from these algorithms. The current work will reduce the processing time by using XMI based activity diagram and optimize the best test path without dependence on others.

5. Conclusion

New technique for the generation of test path from UML diagrams needs to be identified. Also, the study shows that most of existing automation tool using MBT are performed step by step. (e.g. constructing dependency table, creating dependency graph, generating possible paths). The efficient and effective approaches are needed still although there are many existing automatic test case generation approaches. In future, we have planned to generate a test path method that minimizes size of tests, time and cost, and reduce steps at the evolution of generated test paths.

Table 1. Comparison of search optimization algorithms

Algorithms	Time	Complexity	Guarantee	Remark
Tabu	short	low	no	Depend on assigned weight value
GA	mode	high	no	Depend on number of mutation /
	rate			c
ACO	long	very high	yes	D

References

[1] A. Bertolino, "Chapter 5: Software Testing", in IEEE SWEBOK Trial Version 1.00, May 2001.

[2] A.V.K. Shanthi, G. MohanKumar, "A Novel Approach for Automated Test Path Generation using TABU Search Algorithm", International Journal of Computer Applications (0975 – 888)Volume 48– No.13, June 2012.

[3] A.V.K. Shanthi, G. MohanKumar, "A Heuristic Technique for Automated Test Cases Generation from UML Activity Diagram", Journal of Computer Science and Applications. Volume 4, Number 2 (2012), pp. 75-86.

[4] "Certified Tester Foundation Level Syllabus", Released Version 2011, International Software Testing Qualifications Board.

[5] Chanda Chouhan Vivek Shrivastava Parminder S Sodhi, "Test Case Generation based on Activity Diagram for Mobile Application", International

Journal of Computer Applications (0975 – 8887)Volume 57– No.23, November 2012.
[6] Manoj Kumar, Arun Sharma, Rajesh Kumar, " Soft Computing-Based Software Test Cases Optimization: a Survey", International Review on Computers & Software;Jul2011, Vol. 6 Issue 4, p512.
[7] Pakinam N. Boghdady, Nagwa L. Badr, Mohamed Hashem and Mohamed F.Tolba, "A Proposed Test Case Generation Technique Based on Activity Diagrams", International Journal of Engineering & Technology IJET-IJENS Vol: 11 No: 03.

[8] Saurabh Srivastava, Sarvesh Kumar and Ajeet Kumar Verma, "OPTIMAL PATH SEQUENCING IN BASIS PATH TESTING", International Journal of Advanced Computational Engineering and Networking, ISSN (PRINT): 2320-2106, Volume – 1, Issue – 1, 2013.
[9] S. Shanmuga Priya, and Dr. P. D. Sheba, "TEST CASE GENERATION FROM UML MODELS – A SURVEY", International Conference on Information Systems and Computing (ICISC-2013), INDIA, Volume 3, Special Issue 1, January 2013.
[10] V.Mary Sumalatha and Dr G.S.V.P.Raju, "An Model Based Test Case Generation Technique Using Genetic Algorithms", The International Journal of Computer Science & Applications (TIJCSA) Volume 1, No. 9, November 2012 ISSN – 2278-1080.