Software Coverage Analysis: Black Box Approach Using ANT System

Praveen Ranjan Srivastva, Birla Institute of Technology and Science Pilani, India
Saurav Singh Naruka, Birla Institute of Technology and Science Pilani, India
Afaque Alam, Birla Institute of Technology and Science Pilani, India
Nikhil Agarwal, Birla Institute of Technology and Science Pilani, India
Vaibhav Mukeshkumar Shah, Birla Institute of Technology and Science Pilani, India

ABSTRACT

Requirements of the desired software product can be translated into state transition diagram or other UML diagrams. To verify the complete coverage of software requirements, the proposed Ant based approach generates non-repetitive transitions from the input state diagram. This approach has less redundant transitions and also gives uncovered transition in successive paths instead of giving whole redundant path again and again. The paper also contains a comparison between already existing approaches with respect to some parameters like coverage, redundancy, total number of transitions.

Keywords: Ant Colony Optimization, Genetic Algorithm, Software Coverage, Software Testing, State Machine

1. INTRODUCTION

A burning issue for software developing organizations is to deliver the software within a specified time to the client, while maintaining appropriate profit. Testing process is estimated to consume 50% of the total cost involved in software development (Ramlerand & Wolfmaier, 2006). The software needs to be thoroughly tested in order to detect the errors and fix those (Mathur, 2007). Due to limited time and budget it is not feasible to carry out the process of generating test cases, testing and verification, manually (Sommerville, 2009). Thus, automation plays an inevitable role in software testing (Pressman, 2005). Now a day Artificial Intelligence technique is changing the nature of existing automation-testing process (McMinn, 2005). The use of Artificial Intelligence (AI) techniques in Software Engineering are an emerging area of research in which two different domains merge to gain benefit (Briand, 2002; Pedrycz & Petersand, 1998). People have started to review the published works in area of AI in order to extend its application domain in Software Engineering. There are many approaches to apply AI in software testing like particle swarm optimization (Zhang et al., 2005), Bee Colony Optimization, Genetic Algorithm, Ant Colony Optimization, etc.
optimization (Pham et al., 2006), Ant colony optimization (Dorigo & Stutzle, 2005; Ayari et al., 2007) and genetic algorithm (Doungsa-ard et al., 2008). They all are known as meta-heuristic approaches (Srivastava et al., 2008). Genetic algorithm (Doungsa-ard et al., 2008) has not been able to give 100% software coverage and also GA are sufficiently complex and may have a tendency to converge towards local optima rather than the global optimum of the problem. Moreover, as stated in (JeyaMala & Mohan, 2009) GA cannot effectively solve problems in which fitness measure is predicted wrongly, as there would be no strength in algorithm to converge to a solution. Owing to such unpredictable behavior of GA, the algorithm fails to promise 100% coverage of state diagrams every time. Bee Colony Optimization algorithm (Jeya Mala & Mohan, 2009) also does not give full coverage of the software due to complexity of the various control parameters. The proposed approach, which is based on ACO (Ant Colony Optimization technique), has proven its strength for achieving full coverage (Srivastava & Baby, 2010). ACO is one of the approaches, which has widely been used and appears quite promising to achieve full coverage. This paper uses ACO (Dorigo & Stutzle, 2005; Ayari et al., 2007) to generate minimum test cases needed to realize full software coverage.

The paper is structured as follows. Section 2 presents the related work in this field. Section 3 presents the proposed algorithm. Section 4 contains analysis of proposed algorithm. Section 5 debriefs the analysis of different techniques with present work. Section 6 concludes the overall work.

2. BACKGROUND WORK

Before handing over the software to the client, the developers must know whether the software works satisfactorily or not. In other words, the product must be tested thoroughly and further care must be taken that all the requirements in the software specification are incorporated and tested well. Thus, software coverage is important as a part of software testing to verify and validate the product. It is very cumbersome and tiring task if it is done manually. To automate this process, AI algorithms (Zhang et al., 2005) are very promising approach in this regard.

The behavior of software can be easily represented by state model, which is a graph representation in mathematical terms. In this regard, Ants can prove to be a possible candidate to trace the states along the edges and thus, cover the entire State chart, giving us suitable paths. Ant Colony Optimization (Dorigo & Stutzle, 2005; Ayari et al., 2007) is probabilistic technique to solve computational hard problems. Huaizhong and Lam (2005) have described the generation of test sequences using UML based State chart diagram which guarantees coverage of all the states, but there is no surety of covering all the edges in the given state diagram (Huaizhong & Lam, 2005). Thus, the work does not give assurance of full software coverage. (Srivastva et al., 2009) have used ant colony optimization algorithm to find out different possible paths in a control flow graph and the associated strength of each path. It ensures full path coverage in a control flow graph. However, it gives some of the paths with good amount of redundancy in it. Srivastava et al. (2008) have used ACO (Dorigo & Stutzle, 2005; Ayari et al., 2007) algorithm for generation of test data with the help of meta-heuristic approach. They have used GA and ACO algorithm to get test data, but getting successful test data is around 50% only. Doungsa and Hossain (2008) have proposed the automated test data generation using GA from UML test data diagram, but the flaw lies in the chromosome length of fitness function which eventually restricts the full coverage of all transitions. Srivastava and Baby (2010) have developed the full state diagram coverage including all transition based on ACO. In this approach to achieve full software coverage, there is good amount of redundancy being generated on account of repetitive iterations. Mala and Mohan have developed test suite optimization based on artificial Bee Colony Optimization (Mala Jeya & Mohan, 2009). It helps to prioritize the test cases based on how much software is covered by it, but this technique never gives
A Machine Learning Based Meta-Scheduler for Multi-Core Processors
www.igi-global.com/article/machine-learning-based-meta-scheduler/48262?camid=4v1a