CS569 – Milestone#2 Report

Xu Zheng

Onid: zhengxu

Student ID: 932-509-227

May 18, 2016

1. Introduction

In my proposal, my idea was to develop a test case generation algorithm that based on "Feedback-Controlled Random Test Generation" and "Adaptive Random Testing". And the basic idea is selecting the test cases to improve the effectiveness of testing. Since I am still struggling to understand how to use TSTL API, I think it is hard to me to implement my idea using TSTL API. So I want to make some changes to my previous algorithm.

The original idea is that to use multiple pools concurrently to generate the inputs, instead of using a single pool. To be more specific, each pool has different contents, so that different pools will guide the generation toward different direction. By using different pools concurrently, the algorithm can dynamic determine in which direction to continue generate inputs by analyzing the feedback information.

Consider that using multiple pools concurrently is hard to implement, I want to change it to use two pools in order. It means every single pool is generated from previous pool. And there will be some filters used to select inputs from previous pool to current pool. My new idea is inspired by coverage tester algorithm that divides test cases generating into different phase based on threshold coverage.

2. Project Implementation

I have implemented my algorithms. And I have tested my tester with the sut.py generated with avl.tstl.

My tester can read several arguments from command line: timeout, seed, depth, width, faults, coverage and running. To implement this, I used a python library called "argparse" to parse the command line arguments. And I have set defaults for all the arguments mentioned above.

To be more specific, user can set the running time limit, which refers to "timeout" argument, for the tester. And user can set the seed of Random.random() function for random number generation in the code. User also can choose the depth and width of the test. In addition to this, my tester can display how many faults have been encountered in the test, the coverage information and running information by setting the relative arguments to "1" instead of "0".

There are two pools will be created to store the test cases generated. And the time for each one is half the total run time. It means that the first pool will be created during the first half time. And at the rest time, the tester will generate new test cases based on the first pool. The results I got show that tester2 has better performance than tester1.

3. Future Work

Since I have applied my algorithm to the tester, I want to figure out how to improve it. The key point is how to decide which test cases should be selected. In another word, I will work on improving the filters to make the tester to generate test cases effectively. Instead of choosing coverage as the threshold to filter the inputs, maybe I could find another value to determine which inputs should be selected to next pool. I will focus my work on all mentioned above in the future work.