Adaptive Random Testing Generation

Tso-Liang Wu

April 18, 2016

Introduction

According to the lecture material, we know that Random Testing has its advantages. Basically, it is a very effective and intuitive way to test a program or functions. By uniformly choosing test cases from operational profiles, the random testing mostly has good features of reliability and statistical estimates. In random testing, we only care about the rate of faille-causing inputs and take it as the most important factor of the effectiveness measurement. However, according to T.Y. Chen, H. Leung and I.K. Mak's paper "Adaptive Random Testing," we found that the geometric pattern of the failure-causing inputs is also another important factor when considering the testing performance.

According to the research, the mainly differences between adaptive random testing and ordinary random testing are uniform distribution and without replacement. These features not only give adaptive random testing easier mathematical model for analyzing, but also provide testing result which is closer to the reality. Therefore, adaptive random testing is believed that providing a better random testing method which has up to 50% better than traditional random testings, so this is also the reason why I decided to pick adaptive random testing as my project topic.

By classifying the patterns of failure-causing inputs to threes different types (which are point, strip and block), we can modify the ordinary random testing to adapt random testing and get closer inspection. Because we are not solely take the testing input as alone points in adaptive random testing, now we can judge the following test case is humble or not by observing the range or distance of testing inputs. In other word, in adaptive random testing, we should avoid picking neighbor test cases continuously, so that we can avoid meaningless random testing, and improved the testing performance efficiently. To implement this concept, we can easily separate test cases to multiple different groups, and then pick the best case from each group to test the program.

The research mentions a specific way to implement adaptive random testing, which I think is a good method that I can also implement in my project. We can separate two disjoint sets for random testing, which are executed set and candidate set. Firstly, we randomly pick a test case from input domain without replacement and put into a set, called candidate set. Then, we pick the farthest away test case from candidate set as our next testing case. If we didn't reveal any failure, then we put this test case into another set, called executed set.

In order to find the farthest away test case in candidate set, I should clearly define a measurement to calculate the distance in my project. In addition, I should also figure out how to construct the candidate set for adapt random testing.

Plan

- April 20 - April 24: Do more research about adaptive random testing, and some other similar random testing methods.
- April 25 - May 3: Work on the first version of novel test generation. This also includes implement the algorithm in TSTL API.
- May 3 - May 18: Improve the first version of test generation program. In this part, I should fix all bugs from last version and enhance the performance if possible. In addition, I should also vary the parameters and SUTs.
- May 19 - Jun 6: Finalize the project, which includes the final version of novel test generation and all necessary documents.

References

1. Chen, T.Y., Leung H, Mak I.K.: *Adaptive Radom Testing*