Apply Genetic Algorithm on Testing in TSTL

Hao Liu
CS569 Project Proposal
4/19/2016

1. Background Information

Testing can be divided into two different types, functional (black box) and structural (white box) testing, where functional testing is based on functional requirements. Random testing is a common black box testing method which has been already implemented in TSTL. A random test-data generator selects random inputs for the test data from some distribution. This is known as automatic test case. Recently, lot of work is being done for test cases generation using soft computing techniques like fuzzy logic, neural networks, GA, genetic programming and evolutionary computation providing keys to the problem areas of software testing. [5]

GA is well known form of the evolutionary algorithms conceived by John Holland in United States during late sixties. In the past, evolutionary algorithms have been applied in many real life problems. GA is one such evolutionary algorithm. GA has emerged as a practical, robust optimization technique and search method. A GA is a search algorithm that is inspired by the way nature evolves species using natural selection of the fittest individuals. [4] A genetic algorithm typically begins with a random population of solutions and, though a recombination process and mutation operations, gradually evolves the population toward an optimal solution. Obtain an optimal solution is not guaranteed – the challenge is to design the process to maximize the probability of obtaining such a solution. The first step is the selection of the solutions in the current population that will serve as parents in the next generation of solution. This selection requires that the solutions be evaluated for their fitness as parents: solutions that are closer to an optimal solution are judged higher, or more fit, than others.

After solutions have been evaluated, several are selected in a manner that is biased toward the solutions with higher fitness values. The reason for the bias is that a good solution is assumed to be com posed of good components. The resulting solutions from the new population, and the cycle is repeated.

2. Plan

In this project, I would like to implement this algorithm in TSTL. There are two methods from P.R. Srivastava and Tai's paper.

One is identifying the most critical path clusters in a program. [2] The SUT is converted into a CFG. Weights are assigned to the edges of the CFG by applying 80-20 rule. 80 percentage of weight of incoming credit is given to loops and branches and the remaining 20 percentage of incoming credit is given to the edges in sequential path.

Another is based on path coverage testing. The test data is generated for Resource Request algorithm using Ant Colony Optimization algorithm (ACO) and GA. Resource request algorithm is deadlock avoidance algorithm used for resource allocation by operating system

to the processes in execution cycle. [3] If I have time, I will try both method and compare the performance. The minimum target is successfully implementing one method.

References

[1] Goldberg, D.E, Genetic Algorithms: in search, optimization and machine learning, Addison Wesley, M.A, 1989.

[2] Praveen Ranjan Srivastava and Tai-hoon Kim,"Application of genetic algorithm in software testing" International Journal of software Engineering and its Applications, 3(4), 2009, pp.87 – 96

[3] Jose Carlos et. al., "A strategy for evaluating feasible and unfeasible test cases for the evolutionary testing of objectoriented software", AST' 08. ACM, 2008, http://www.cs.bham.ac.uk/~wbl/biblio/cache/http___jcbri beiro.googlepages.com_ast12-ribeiro.pdf, Accessed on 6.11.2012.

[4] Sharma C, Sabharwal S, Sibal R. A survey on software testing techniques using genetic algorithm[J]. arXiv preprint arXiv:1411.1154, 2014.

[5] Pargas R P, Harrold M J, Peck R R. Test-data generation using genetic algorithms[J]. Software Testing Verification and Reliability, 1999, 9(4): 263-282.