Advanced Software Testing and Reverse Engineering Lab 3

INTRODUCTION

Evolutionary algorithms (EAs) are population-based algorithms that are widely used to generate software inputs or tests in an automated fashion.

EAs methods use a guided random process for constructing new inputs/tests that try to trigger new code branches. They recombine previously tried inputs/tests guided by a heuristic that measures the proximity to conditions that trigger new branches (approach level and branch distance).

In this Lab, you will construct and experiment EAs, understand how they work, learn how to use them, and discover their strengths and weaknesses.

LEARNING OUTCOMES

After completing this assignment, you will be able to:

- *Build genetic algorithms for generating new test inputs.*
- *Use and understand modern population-based software testing tools such as EvoSuite (http://www.evosuite.org/).*
- *Empirically compare EAs algorithms and tools*

INSTRUCTIONS

This assignment continues with the code you wrote for Lab assignment 1. You can work on any problem from the RERS challenge (reachability analysis).

Task 1: population-based search

Your job is to extend your hill-climber from Lab 1 to a basic genetic algorithm. Instead of starting from a random trace and improving it one step at a time, you have to keep a population of traces in memory and improve them using cross-over and mutation:

- For cross-over, you should treat a trace as a genome and cut them at random.
- Mutation is a random change to the genome (a random hill-climbing step).
- Implement the tournament selection strategy as explained in the lecture.
- Use as search heuristic the branch distances only
- As you have multiple targets (branches at the same time) you have to choose a strategy to handle all branches. From the lecture, there are three main approaches: single-target approach, whole-suite approach, and the many-objective approach. You are free to choose which of these three approaches to implement.

Experiment with different crossover and mutation rates and show the performance of your genetic algorithm in terms of branches covered in a convergence plot. The convergence plot depicts the number of branches (y-axis) covered over time (x-axis)

Task 2: population-based testing tools

In this task, you will use EvoSuite, which is the state-of-the-art test case generation. EvoSuite implements all three population-based approaches to handle all branches, i.e., single-target, whole suite, and many-objective. To complete this task, you need to:

- Compile the RERS problems (EvoSuite does bytecode-level instrumentation)
- Run EvoSuite on the same RERS problems used for Task 1
- Compare the performance of the population-based method you implemented in Task 1 with the results produced by EvoSuite.
- To measure the performance, you can use either branch or statement coverage (higher coverage is better)

Instructions on how to run EvoSuite on RERS problems are posted on Brightspace. To allow a fair comparison, you should consider the two main aspects:

1) You should run EvoSuite and your tool (from Task 1) with the same search budget (e.g., 5 min). EvoSuite already has an input parameter to set up the search budget. You should implement a timeout for the search in your tool too.

2) EAs are randomized (you can see them as variants of random fuzzers). As such, they can return different results when executed multiple times. To properly

compare EAs (i.e., your tool and EvoSuite), you have to run them at least 10 times and compare the average (arithmetic mean) results.

RESOURCES

Slides from Lectures 2, 3

Study:

1. A detailed investigation of the effectiveness of whole test suite generation http://www.evosuite.org/wp-content/papercite-data/pdf/emse16_effectiveness.pdf
2. Reformulating branch coverage as a many-objective optimization problem https://selab.fbk.eu/kifetew/papers/icst15.pdf
3. Disposable testing: avoiding maintenance of generated unit tests by throwing them away https://dl.acm.org/citation.cfm?id=3098414

First read the following papers, and familiarize yourself with the EvoSuite (http://www.evosuite.org) tool.

All code samples are available on GitHub https://github.com/tudelft-cs4110-2019

Detailed instructions on how to run EvoSuite on RERS are posted on Brightspace.

PRODUCTS

A small report of max 2 A4 pages answering the questions from the 2 tasks, visualizations are allowed in at most 2 extra A4 pages.

An archive (tar/zip) containing the code for computing the results.

ASSESSMENT CRITERIA

You can either pass or fail this assignment. We expect everyone to pass. You have to complete all lab assignments. Submissions of which the report text does not fit into 2 A4s will not be evaluated.

Your work will be evaluated based on its completeness (having done all tasks), the correctness of the implementation, and demonstration that you understand the results in the analysis. When deemed insufficient, you will receive feedback and will be given a one-week grace period to fix any shortcomings.

SUPERVISION AND HELP

There will be lab sessions every Wednesday, where the teachers and TA will be available to answer any questions you may have. The preferred way to ask questions is through Mattermost.

SUBMISSION AND FEEDBACK
The submission is through Brightspace. You will receive feedback within one week after the deadline.