Evaluation of the Mutation Testing Approaches used in Mutatest and MutPy on Open-Source Python Programs

Blake Davis
CS 514 Term Project

Introduction

This project has 2 main goals:

1. To evaluate the relationship between mutation score and statement coverage percentage.
2. To evaluate each mutation testing approach in terms of cost-effectiveness.

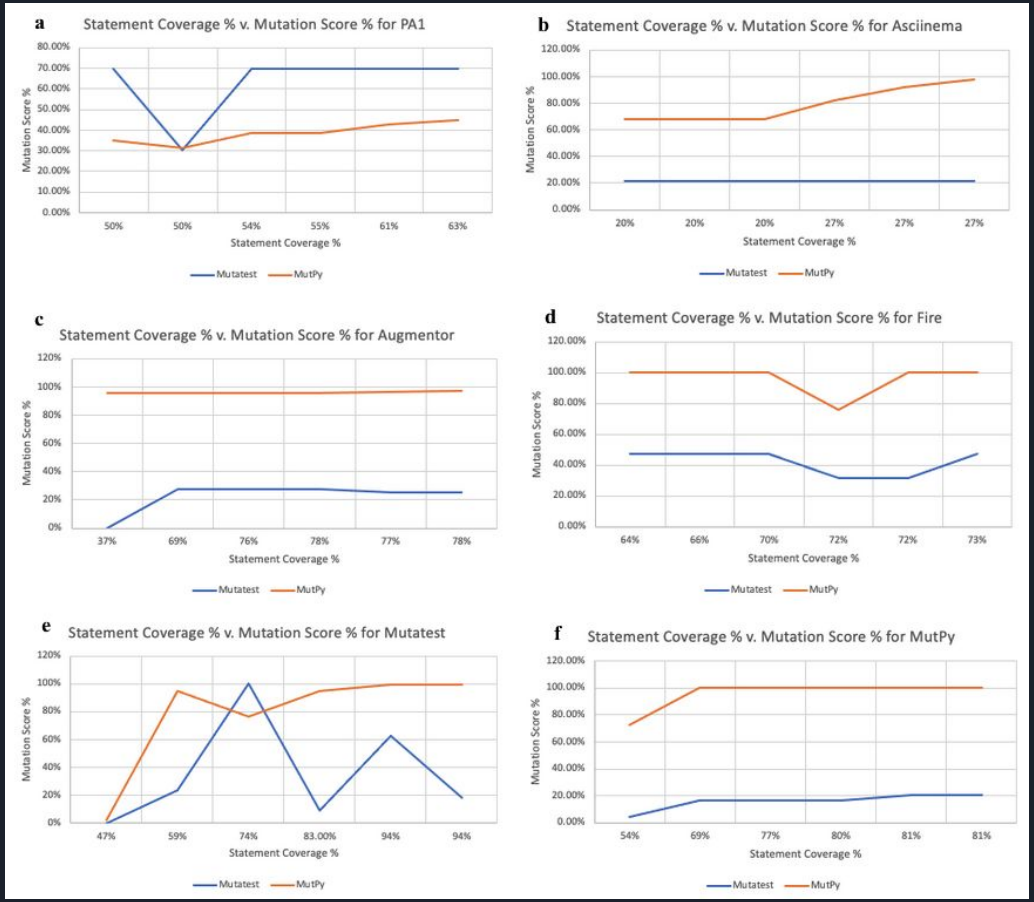
Study Design:

- 2 Mutation Testing Tools for Python: Mutatest and MutPy
- 6 Python Programs: CS 220 Program + 5 Open-Source Programs
- 6 Test Suites Per Python Program

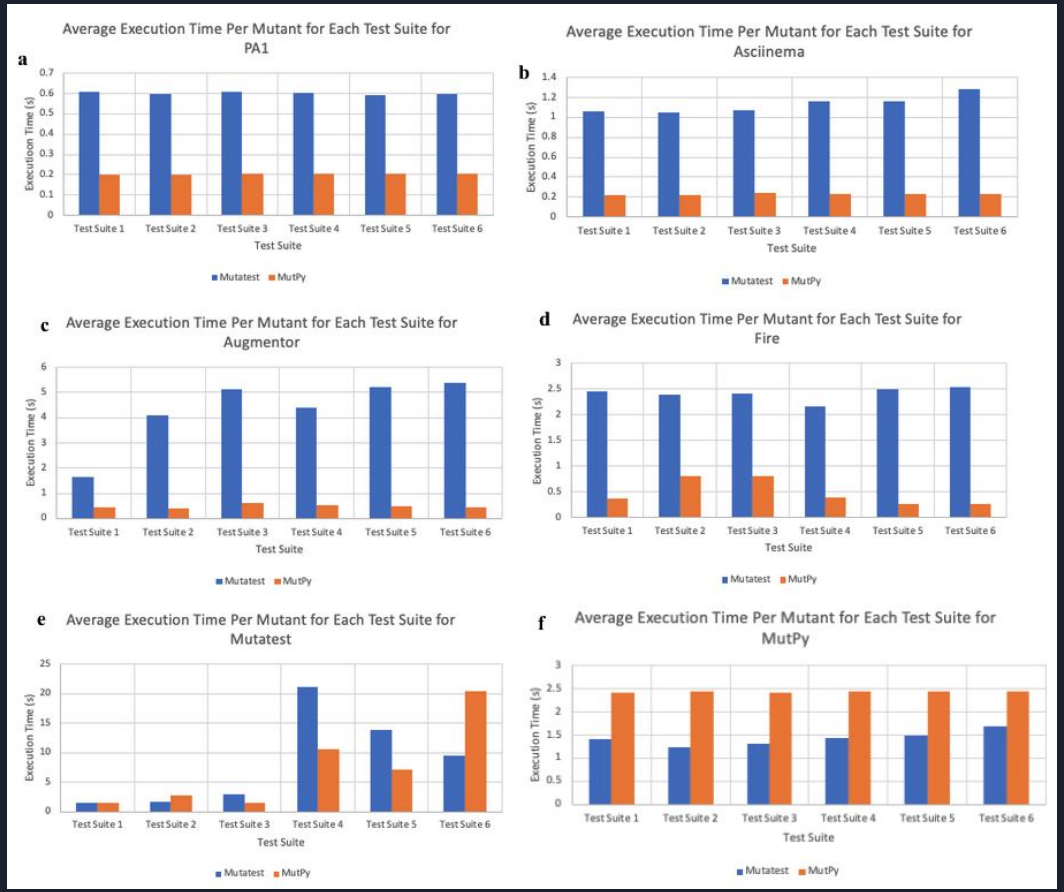
Metrics:

- Mutation Score
- Statement Coverage Percentage
- Execution Time
- Average Execution Time Per Mutant
- Percentage of Program Mutated

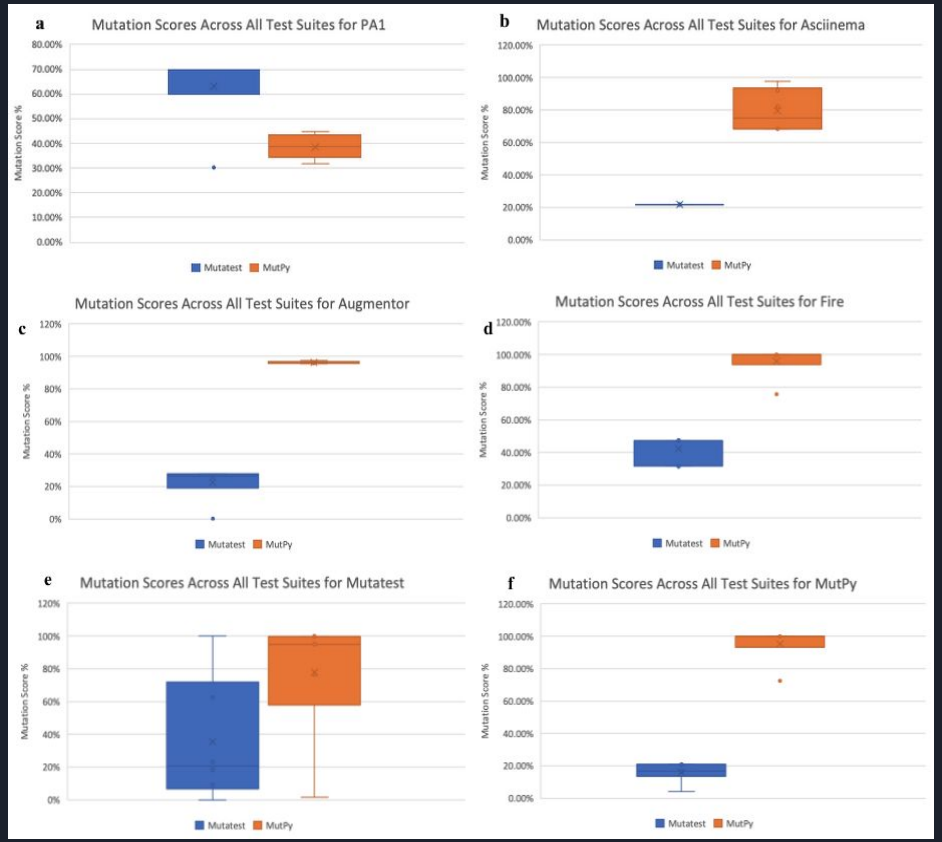
Results - Page 1



Results - Page 2



Results - Page 3



Conclusions

- Statement Coverage Percentage is not strongly correlated with Mutation Score.
  - Higher Statement Coverage may mean more tests which may affect Mutation Score.
  - Related Work supports this.
- On average, Mutatest takes longer per mutant with a lower execution time.
  - Executes Less Mutants Per Execution
  - Overhead of Running in Parallel?
- MutPy takes longer on the source code of Mutatest and MutPy
  - Future Research: Why? What causes MutPy to slow down on these programs?

- Related Work gave some new perspectives on Mutation Testing
  - Google only runs mutation testing on code that is currently being reviewed