Empirical Evaluation of Test Suite Reduction

Adrian Regenfuß

December 16, 2020

Contents

Abstract

As a response to ever-growing test suites with long runtimes, several different approaches have been developed to reduce the time for test suite execution to give useful results. One of these approaches is test suite reduction: selecting a sample of tests that maximizes coverage on the tested source code. This paper attempts to replicate the findings in Cruciani et al. 2019, which borrows techniques from big data to handle very large test suites. We use independently generated testing data from open source projects, and find that TODO.

1 Introduction

3 pages

To prevent the introduction (or re-introduction) of bugs, software developers often test their software after making changes to it. This is known as regression testing, and takes up a significant portion of development cost. However, the resulting test suites can grow quite significantly in size and execution time, which hinders development speed and increases costs.

This work attempts to determine how different test suite reduction strategies compare to each other in terms of time performance, fault detection loss and magnitude of reduction.

2 Related Work

5 pages

2.1 Handling Large Test Suites

2.1.1 Test Case Selection

2.1.2 Test Case Priorization

2.1.3 Test Suite Reduction

2.2 Algorithms for Reduction

2.2.1 Greedy Selection

Needs coverage information

2.2.2 Clustering

2.2.3 Searching

2.2.4 Hybrids

3 Approach

8 pages

3.1 Replicating "Scalable Approaches for Test Suite Reduction"

1 page

3.2 Algorithms Used

7 pages

3.2.1 FAST

4 pages

FAST++

FAST-all

FAST-CS

FAST-pw

3.2.2 Adaptive Random Testing

2 pages

ART-D

ART-F

3.2.3 Greedy Algorithm

$\frac{1}{2}$ a page

3.2.4 Random Selection

$\frac{1}{2}$ a page
New method, after Khan et al. 2018, p. 17

4 Evaluation

17 pages

4.1 Research Questions

Research Question 1: Does the Relative Effectiveness of the Different Algorithms Replicate the Results in Cruciani et al. 2019?

Research Question 1.1: Does Their Relative Effectiveness in TSR Replicate the Findings in Cruciani et al. 2019?

Research Question 1.2: Does Their Relative Effectiveness in FDL Replicate the Findings in Cruciani et al. 2019?

Research Question 2: Does the Relative Runtime Performance of the Different Algorithms Replicate the Results in Cruciani et al. 2019?

Research Question 3: How Much Better Than Random Selection are Specialized Algorithms?

(Possibly) Research Question 4: Do the Results in Cruciani et al. 2019 Replicate with the Original Test Data?

4.2 Study Setup

4.3 Study Objects

4.3.1 Selecting Projects

Medium-Sized Java Projects
assertj-core, commons-lang, commons-math, commons-collections, jopt-simple, jsoup

4.3.2 Generating Coverage Information

Only Line Coverage
Using Teamscale Jacoco Agent

4.3.3 Collecting Fault Coverage Information

Using Pitest

4.3.4 Combining Tests Suites

4.4 Results

Research Question 1.1

Research Question 1.2

Research Question 2

Research Question 3

(Possibly) Research Question 4

References

Emilio Cruciani, Breno Miranda, Roberto Verdecchia, and Antonia Bertolino. Scalable approaches for test suite reduction. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 419–429. IEEE, 2019.

Saif Ur Rehman Khan, Sai Peck Lee, Nadeem Javaid, and Wadood Abdul. A systematic review on test suite reduction: Approaches, experiment's quality evaluation, and guidelines. *IEEE Access*, 6:11816–11841, 2018.