DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Empirical Evaluation of Test Suite Reduction

Adrian Regenfuß

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Empirical Evaluation of Test Suite Reduction

Empirische Evaluation von Test-Suiten Reduktion

Author:	Adrian Regenfuß
Supervisor:	Prof. Dr. Dr. h.c. Manfred Broy
Advisor:	Dr. Elmar Jürgens, Raphael Nömmer, Roman Haas
Submission Date:	Submission date

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, Submission date Adrian Regenfuß

Acknowledgments

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 [Cru+19], 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.

Contents

1 Introduction

3 pages

Software often has has faults: ways in which the actual behavior of the software diverges from the intended or specifie behavior. Computer scientists have devised different strategies for finding and removing bugs: Formal software verification, code reviews, and different types of testing: unit testing, which tests the behavior of small software modules (such as classes), integration testing, which shows how well the software works in the environment it is used in, and regression testing, which shows developers whether they have introduced new bugs after the last change to the software.

Regression testing takes up a significant portion of development cost, and the resulting test suites can grow quite significantly in size and execution time, which hinders development speed and increases costs.

To mitigate the costs and runtimes of regression test suites, different strategies have been devised: test case selection, which selects a subset of tests for the current execution of tests, test suite reduction, which permanently deletes a subset of tests from the test suite, and test case priorization, which changes the order of test execution to maximize the amount of faults that is found early in the test suite execution.

[Cru+19] presents a new family of test suite reduction algorithms, called the FAST algorithms (first developed in [Mir+18]), and compares them to the algorithms presented in [Che+10], as well as the greedy algorithm presented in [Rot+01]. They implement 4 algorithms from the FAST family, 2 from the ART family and the greedy algorithm, and compare the different algorithms on 10 different test programs and their test suites. They compare the performance of the test suite reduction algorithms on 3 different variables: test suite reduction, fault detection loss, and runtime.

This work attempts to replicate their findings using the data from 6 additional projects, as well as adding random test case selection as a baseline test suite reduction method to compare the other methods to (following **Recommendation 8** from [Kha+18]).

This work then 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 Terms and Definitions

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

3 Related Work

5 pages

3.1 Test Case Selection

3.2 Test Case Priorization

3.3 Test Suite Reduction

3.3.1 Greedy Selection

Needs coverage information

3.3.2 Clustering

3.3.3 Searching

3.3.4 Hybrids

4 Approach

8 pages

4.1 Replicating ”Scalable Approaches for Test Suite Reduction”

1 page

4.2 Implemented Algorithms

7 pages

4.2.1 FAST

4 pages

FAST++

FAST-all

FAST-CS

FAST-pw

4.2.2 Random Selection

$\frac{1}{2}$ a page
New method, after [Kha+18] (p. 17)

4.2.3 Adaptive Random Testing

2 pages

ART-D

ART-F

4.2.4 Greedy Algorithm

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

5 Evaluation

17 pages

5.1 Research Questions

Research Question 1: Does the Relative Effectiveness of the Different Algorithms Replicate the Results in [Cru+19]?

Research Question 1.1: Does Their Relative Effectiveness in TSR Replicate the Findings in [Cru+19]?

Research Question 1.2: Does Their Relative Effectiveness in FDL Replicate the Findings in [Cru+19]?

Research Question 2: Does the Relative Runtime Performance of the Different Algorithms Replicate the Results in [Cru+19]?

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

(Possibly) Research Question 4: Do the Results in [Cru+19] Replicate with the Original Test Data?

5.2 Study Design

5.3 Study Objects

5.4 Selecting Projects

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

5.5 Study Setup

5.5.1 Generating Coverage Information

Only Line Coverage
Using Teamscale Jacoco Agent

5.5.2 Collecting Fault Coverage Information

Using Pitest

5.5.3 Combining Tests Suites

5.6 Results

5.6.1 Research Questions

Research Question 1.1

Research Question 1.2

Research Question 2

Research Question 3

(Possibly) Research Question 4

5.6.2 Running Time

5.7 Discussion

Comparison to ”Scalable Approaches to Test Suite Reduction”

5.8 Threats to Validity

5.8.1 Conclusion Validity

5.8.2 Internal Validity

5.8.3 Construct Validity

5.8.4 External Validity

6 Future Work

1 page

7 Summary

2 pages

List of Figures

List of Tables

Bibliography

[Che+10] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. Tse. "Adaptive random testing: The art of test case diversity." In: *Journal of Systems and Software* 83.1 (2010), pp. 60–66.

[Cru+19] E. Cruciani, B. Miranda, R. Verdecchia, and A. Bertolino. "Scalable approaches for test suite reduction." In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE. 2019, pp. 419–429.

[Kha+18] S. U. R. Khan, S. P. Lee, N. Javaid, and W. Abdul. "A systematic review on test suite reduction: Approaches, experiment's quality evaluation, and guidelines." In: *IEEE Access* 6 (2018), pp. 11816–11841.

[Mir+18] B. Miranda, E. Cruciani, R. Verdecchia, and A. Bertolino. "Fast approaches to scalable similarity-based test case prioritization." In: *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE. 2018, pp. 222–232.

[Rot+01] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. "Prioritizing test cases for regression testing." In: *IEEE Transactions on software engineering* 27.10 (2001), pp. 929–948.