Using Synthetic Test Suites to Empirically Compare Search-Based and Greedy Prioritizers

Zachary Williams
Gregory M. Kapfhammer

Department of Computer Science
Allegheny College
http://www.cs.allegheny.edu/

Genetic and Evolutionary Computation Conference
Late Breaking Abstract Workshop
July 2010
Important Contributions

Synthetic Test Suites

Detailed Empirical Study

Use **synthetic test suites** to empirically evaluate the **efficiency** and **effectiveness** of search-based and greedy prioritizers.
Use synthetic test suites to empirically evaluate the efficiency and effectiveness of search-based and greedy prioritizers.
Important Contributions

Synthetic Test Suites

Detailed Empirical Study

Use synthetic test suites to empirically evaluate the efficiency and effectiveness of search-based and greedy prioritizers.
Use **synthetic test suites** to empirically evaluate the **efficiency and effectiveness** of search-based and greedy prioritizers.
Correct programing defect
Correct programing defect
Overview of Regression Testing

Correct programing defect
Overview of Regression Testing

Correct programing defect
Overview of Regression Testing

Correct programming defect
Overview of Regression Testing

Correct programing defect
Overview of Regression Testing

Add new functionality
Overview of Regression Testing

Add new functionality
Overview of Regression Testing

Modify test suite
Overview of Regression Testing

Modify test suite
Complete retesting is often prohibitively expensive
Requirements necessitate the coverage of the state and/or structure of a program under test.
Requirements necessitate the coverage of the **state** and/or **structure** of a program under test.
Requirements necessitate the coverage of the **state** and/or **structure** of a program under test.
Regression Test Suite Prioritization

Each test covers specific **requirements** in a certain amount of **time** and thus the **ordering** is critical.
Each test covers specific requirements in a certain amount of time and thus the ordering is critical.
Prioritized test suites cover requirements faster thus enabling the rapid detection of defects.
Prioritized test suites cover requirements faster thus enabling the rapid detection of defects.
Prioritized test suites cover requirements faster thus enabling the rapid detection of defects.
Prioritized test suites cover requirements faster thus enabling the rapid detection of defects.
Prioritized test suites cover requirements faster thus enabling the rapid detection of defects.
Prioritized test suites cover requirements faster thus enabling the rapid detection of defects
Prioritized test suites cover requirements faster thus enabling the rapid detection of defects.
Regression Test Suite Prioritization

Prioritized test suites cover requirements faster thus enabling the rapid detection of defects.
Prioritized test suites cover requirements faster thus enabling the rapid detection of defects
Testers can use greedy (Rothermel et al. TSE 2001) and search-based (Li et al. TSE 2007) methods to reorder suites.
Regression Test Suite Prioritization

QUESTION: Which prioritization technique is the best?
Existing Prioritization Techniques

Greedy approaches select the next best test case

Hill climbers search the state space for improved orderings

Conventional wisdom dictates that greedy generally outperforms hill climbing in terms of both efficiency and effectiveness.
Existing Prioritization Techniques

**Greedy** approaches select the next best test case

**Hill climbers** search the state space for improved orderings

Conventional wisdom dictates that greedy generally outperforms hill climbing in terms of both efficiency and effectiveness
**Existing Prioritization Techniques**

**Greedy** approaches select the next best test case.

**Hill climbers** search the state space for improved orderings.

Conventional wisdom dictates that greedy generally outperforms hill climbing in terms of both efficiency and effectiveness.
Existing Prioritization Techniques

Greedy approaches select the next best test case

Hill climbers search the state space for improved orderings

Conventional wisdom dictates that greedy generally outperforms hill climbing in terms of both efficiency and effectiveness

http://artedi.ebc.uw.se/course/Enbo01/Phylogeny PHYLOGENY.html
Conducting an Empirical Evaluation

Determine Problem

These highlighted tasks are manual, expensive, and prone to error.
Conducting an Empirical Evaluation

Determine Problem

Download Case Studies

These highlighted tasks are **manual, expensive, and prone to error**
Conducting an Empirical Evaluation

These highlighted tasks are manual, expensive, and prone to error.
Conducting an Empirical Evaluation

These highlighted tasks are manual, expensive, and prone to error.
Conducting an Empirical Evaluation

- Determine Problem
- Download Case Studies
- Configure Applications and Test Suites
- Produce Coverage Report
- Conduct Experiment

These highlighted tasks are manual, expensive, and prone to error.
Conducting an Empirical Evaluation

These highlighted tasks are **manual, expensive, and prone to error**
Conducting an Empirical Evaluation

Synthetically generating a test suite is automated, effective, and efficient.
Conducting an Empirical Evaluation

- Determine Problem
- Download Case Studies
- Configure Applications and Test Suites
- Produce Coverage Report
- Conduct Experiment

Generate Synthetic Test Suites

Synthetically generating a test suite is *automated*, *effective*, and *efficient*.
The **total number of tests** controls how many tests the suite will contain.
The **total number of requirements** governs how many requirements the test suite will cover.
The total number of coverage points controls how many unique test-requirement pairs the test suite will contain.
The balancing configuration dictates how the coverage points will be distributed in the synthetic test suite.
Our empirical results show that synthetic generation takes less than 0.2 seconds for extremely large test suites.
Generating Synthetic Test Suite

- Tests
- Requirements
- Coverage Points
- Balancing Approach

Synthetic Test Suite Generator

Synthetic Test Suite (Coverage Report)

Contains information concerning the requirements covered and the execution time of each test
Empirical Results – Prioritizer Efficiency

HC demonstrated to be more efficient than GRD for large test suites.
HC demonstrated to be more efficient than GRD for large test suites
Empirical Results – Prioritizer Effectiveness

As the amount of coverage points in the test suite increases, the performance of HC becomes comparable to that of GRD.
Empirical Results – Prioritizer Effectiveness

As the amount of coverage points in the test suite increases, the performance of HC becomes comparable to that of GRD.
Empirical Results – Prioritizer Effectiveness

The trend is evident over a wide range of experimental configurations.
Empirical Results – Prioritizer Effectiveness

The trend is evident over a wide range of experimental configurations.
Conclusion

- Synthetic test suite generation is efficient
- Enable the identification of fundamental trade-offs

Future Work

- Apply our technique to genetic and other algorithms
- Implement and evaluate new and different synthetic generators
Conclusion and Future Work

Conclusion

- Synthetic test suite generation is efficient
- Enable the identification of fundamental trade-offs

Future Work

- Apply our technique to genetic and other algorithms
- Implement and evaluate new and different synthetic generators

http://www.cs.allegheny.edu/~gkapfham/research/kanonizo/
Conclusion

- Synthetic test suite generation is efficient
- Enable the identification of fundamental trade-offs

Future Work

- Apply our technique to genetic and other algorithms
- Implement and evaluate new and different synthetic generators
Conclusion

- Synthetic test suite generation is efficient
- Enable the identification of fundamental trade-offs

Future Work

- Apply our technique to genetic and other algorithms
- Implement and evaluate new and different synthetic generators

Conclusion and Future Work
Conclusion and Future Work

Conclusion

- Synthetic test suite generation is efficient
- Enable the identification of fundamental trade-offs

Future Work

- Apply our technique to genetic and other algorithms
- Implement and evaluate new and different synthetic generators

http://www.cs.allegheny.edu/~gkapfham/research/kanonizo/
Conclusion and Future Work

**Conclusion**
- Synthetic test suite generation is efficient
- Enable the identification of fundamental trade-offs

**Future Work**
- Apply our technique to genetic and other algorithms
- Implement and evaluate new and different synthetic generators

---

**Using Synthetic Test Suites to Empirically Compare Search-Based and Greedy Priorizers**

Zachary D. Williams  
Department of Computer Science  
Allegheny College  
willaz@allegheny.edu

Gregory M. Kapfhammer  
Department of Computer Science  
Allegheny College  
gkapfham@allegheny.edu

**ABSTRACT**

The increase in the complexity of modern software has led to the common practice of using synthetic test suites to verify the correctness of software. However, using this type of test suite to validate the correctness of software is not always efficient. In this paper, we present two types of synthetic test suite generation methods. We implemented a test suite generation method using search-based test suite generation and a greedy test suite generation method. We then compared the performance of the two methods using a set of real-world applications. The results show that the search-based test suite generation method is more efficient than the greedy test suite generation method.

**Categories and Subject Descriptions**

D.2.3 [Software Engineering]: Testing and Debugging  
General Terms: Experimentation, Performance

**Keywords:** search-based, greedy test suite generation

1. **INTRODUCTION**

Software developers often introduce defects during the implementation process. Regression testing methods establish a correspondence between defects and tests in order to detect defects within a program by running a collection of tests. Since regression testing can be very time-consuming, developers use search-based and greedy prioritization techniques to produce a test ordering that will reveal defects in the shortest time. In this paper, we present two types of synthetic test suite generation methods.

Suppose a test suite $D = \{d_1, d_2, \ldots, d_n\}$ covers the set of requirements $R = \{r_1, r_2, \ldots, r_m\}$. Each test case $d_i \in D$ is associated with the set $R_i \subseteq R$. During the empirical study of search-based and greedy test suite generation, we observe that the search-based test suite generation method is more efficient than the greedy test suite generation method.

---

**Figure 1: Execution Times - Fully Random**

and greedy test suite prioritizers, researchers often use the $T$ and $N(T)$ associated with real-world case study applications. Yet, this practice is difficult and time-consuming because of the need to tailor prototype test ordering tools for complex real-world programs. Furthermore, small case study applications may not be representative of all real-world programs. Thus, this paper describes two simple methods for generating synthetic test suites and demonstrates how they reveal fundamental trade-offs in test prioritization techniques. We used two parameters to create a synthetic test suite: requirement $X$ and coverage-point factor $F$. For a given test suite size, $N$, we control how many requirements are generated for $N \times F$. Here, such that $|T| = N \times |F|$. After setting the size of the test suite and requirement set, we use $F$ to determine the number of times the requirements are covered, denoted $C$, as a fraction of the total number of possible coverage points, so that $T = F \times |T| = |F|$. 

---

Zachary Williams (Allegheny College)  
Synthetic Test Suite Prioritization  
GECCO 2010
Conclusion and Future Work

**Conclusion**

- Synthetic test suite generation is efficient
- Enable the identification of fundamental trade-offs

**Future Work**

- Apply our technique to genetic and other algorithms
- Implement and evaluate new and different synthetic generators

---

Conclusion

- Synthetic test suite generation is efficient
- Enable the identification of fundamental trade-offs

Future Work

- Apply our technique to genetic and other algorithms
- Implement and evaluate new and different synthetic generators

---

**http://www.cs.allegheny.edu/~gkapfham/research/kanonizoz/**