Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing

Gregory M. Kapfhammer†

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

University of Delhi – May 9, 2012

† Joint with Jonathan Miller Kauffman (Allegheny College)
Presenter Introduction: Gregory M. Kapfhammer
Software is Everywhere

Program

Computer
Server
Software is Everywhere

- Program
- Desktop Computer
- Program
- Computer Server
Software is Everywhere

Program

Desktop Computer

Program

Computer Server

Program

Mobile Computer
Software is Everywhere

- Program
  - Desktop Computer
- Program
  - Computer Server
- Program
  - Mobile Computer
- Program
  - Household Appliance
Software is Everywhere

- Program
  - Desktop Computer
  - Scientific Device
- Program
  - Computer Server
  - Household Appliance
- Program
  - Mobile Computer
Software is Everywhere

- Program
- Desktop Computer
- Program
- Computer Server
- Program
- Mobile Computer
- Program
- Scientific Device
- Program
- Household Appliance
- Program
- Network Router
## Software Challenges

### Software is Complex

![Computer Software](Computer Software.png)
Software is Complex

Lines of Code

Computer Software

Kapfhammer

Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Software Challenges

Software is Complex

- Computer
- Software
- Lines of Code
- Numerous Features
Software Challenges

Software is Complex

- Lines of Code
- Numerous Features
- Feature Interactions

Computer Software
Software is Complex

- Computer Software
  - Lines of Code
  - Numerous Features
  - Feature Interactions
  - Execution Environments
Software is Complex

Software entities are more complex for their size than perhaps any other human construct - Frederick P. Brooks, Jr.
Software is Evolving

Program

Execution Environment
Software is Evolving

![Diagram showing the relationship between program and execution environment](image-url)
Software Challenges

Software is Evolving

Program Changed because of the addition of a new feature or the correction of a defect
Software Challenges

Software is Evolving

Program

Execution Environment
Software is Evolving
Software is Evolving

Execution Environment Changed due to an upgrade in a kernel, device driver, or virtual machine
Software Challenges

Regression Testing to the Rescue
Regression Testing to the Rescue

Computer Software

Pervasive

Kapfhammer
Allegheny College
Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
## Software Challenges

### Regression Testing to the Rescue

Software is pervasive and complex, requiring robust testing methods. Regression testing is a critical tool in ensuring software quality improvement through repeated test execution. This approach has been explored extensively in the past and continues to evolve as the field of software testing advances.

![Diagram showing the relationship between Computer Software, Pervasive, and Complex]

---

**Kapfhammer**

Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Regression Testing to the Rescue

Computer Software

Pervasive

Evolving

Complex

Software Challenges

Kapfhammer Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Regression Testing to the Rescue

Regression Testing supports the efficient construction of pervasive software that is complex and rapidly evolving.
## What is a Test Case?

<table>
<thead>
<tr>
<th>Method</th>
<th>Under Test</th>
</tr>
</thead>
</table>

Kapfhammer

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
What is a Test Case?

Input → Method Under Test
What is a Test Case?

Input → Method Under Test → Output
What is a Test Case?

Test Set Up

Method Under Test
What is a Test Case?

Test Set Up

Input

Method Under Test
What is a Test Case?

![Diagram showing the components of a test case: Input, Test Set Up, Method Under Test, Output]
What is a Test Case?

- **Input**
- **Method Under Test**
- **Output**
- **Test Set Up**
- **Test Clean Up**
What is a Test Case?
What is a Test Case?

- **Test Set Up**
- **Method Under Test**
- **Output**
- **Expected Output**
- **Test Oracle**
- **Test Clean Up**

Testing Opportunities

Kapfhammer Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
What is a Test Case?

```
Input    Method Under Test    Output    Expected Output
          Test Set Up            Test Clean Up    Test Oracle
```

Kapfhammer

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
What is a Test Case?

1. Input
2. Method Under Test
3. Test Set Up
4. Test Clean Up
5. Output
6. Expected Output
7. Test Oracle
8. Test Verdict

Kapfhammer

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
What is a Test Case?

The test case passes and the code is correct!
What is a Test Case?

Input → Method Under Test → Output → Test Oracle → Expected Output

- Test Set Up
- Test Clean Up
- Expected Output
- Test Verdict
What is a Test Case?

The test case fails and a defect is found!
What is a Test Suite?

$T_1$
What is a Test Suite?

$T_1 \rightarrow T_2$
What is a Test Suite?

\[ T_1 \rightarrow T_2 \rightarrow T_3 \]
What is a Test Suite?
What is a Test Suite?

\[ T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow \ldots \]
What is a Test Suite?

\[ T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow \ldots \rightarrow T_n \]
What is a Test Suite?

Organize the Test Cases into a Test Suite

$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow \ldots \rightarrow T_n$
What is a Test Suite?

Organize the Test Cases into a Test Suite

$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow \ldots \rightarrow T_n$

Tool Support for Software Testing?
What is a Test Suite?

Organize the Test Cases into a Test Suite

$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow \ldots \rightarrow T_n$

Tool Support for Software Testing?

JUnit
What is a Test Suite?

Organize the Test Cases into a Test Suite

Tool Support for Software Testing?

JUnit

Apache Ant
What is a Test Suite?

Organize the Test Cases into a Test Suite

Tool Support for Software Testing?

JUnit  Apache Ant  Eclipse
Test Suite Management

Organize the Test Cases into a Test Suite

\[ T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow \ldots \rightarrow T_n \]
Test Suite Management

Regression Testing Technique
Test Suite Management

What if Some Test Cases are More Effective?

Regression Testing Technique

Kapfhammer

Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Test Suite Management

What if Some Test Cases are More Effective?

Regression Testing Technique

Prioritization
Test Suite Management

What if Some Test Cases are More Effective?

Regression Testing Technique

Prioritization
Test Suite Management

What if Some Test Cases are Redundant?

Regression Testing Technique

Prioritization

Kapfhammer Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Test Suite Management

What if Some Test Cases are Redundant?

Regression Testing Technique

Prioritization

Reduction
Test Suite Management

What if Some Test Cases are Redundant?

Regression Testing Technique

Prioritization

Reduction
Test Suite Management

What if Only Certain Tests are Needed?

Regression Testing Technique

Prioritization  Reduction
Test Suite Management

What if Only Certain Tests are Needed?

Regression Testing Technique

Prioritization  Reduction  Selection
Model of Regression Testing
Model of Regression Testing

Supporting Methods

Start → Coverage Report
Model of Regression Testing

![Diagram showing the model of regression testing]

- Start
- Coverage Report
- Selection
- Reduction
- Prioritization
Model of Regression Testing

Start → Coverage Report → Original Test Suite

- Selection
- Reduction
- Prioritization
Model of Regression Testing
Model of Regression Testing

Start → Coverage Report → Selection → Reduction → Prioritization → Modified Test Suite → Test Suite Execution

Supporting Methods

Kapfhammer
Allegheny College
Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Model of Regression Testing

Supporting Methods

Kapfhammer

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Supporting Methods

Model of Regression Testing

Start → Coverage Report → Original Test Suite → Selection → Reduction → Prioritization → Modified Test Suite → Program → Adequacy Criterion → Test Suite Execution → Test Coverage Monitoring → Test Results
Supporting Methods

Model of Regression Testing

- Start
- Coverage Report
- Original Test Suite
  - Selection
  - Reduction
  - Prioritization
- Modified Test Suite
- Test Suite Execution
  - Program
  - Adequacy Criterion
  - Test Coverage Monitoring
- Test Results
- End

Kapfhammer
Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Model of Regression Testing

Use the Coverage Report During the Next Round of Regression Testing
Model of Regression Testing

Use the Same Test Suite for the Next Round of Regression Testing
Model of Regression Testing

Make a New Test Suite for the Next Round of Regression Testing
Model of Regression Testing

Make a New Test Suite for the Next Round of Regression Testing

Our Tools Support All of the Phases in this Model!
Supporting Methods

Test Suite Adequacy

$T_1$  $T_2$
Introduction
Software Testing
Regression Testing
Empirical Evaluation
Conclusion

Supporting Methods

Test Suite Adequacy

$T_1$  $T_2$  $T_3$  $T_4$
Test Suite Adequacy

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$
Test Suite Adequacy

\[ T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6 \quad T_7 \quad T_8 \]
Test Suite Adequacy

$T_1$  $T_2$  $T_3$  $T_4$  $T_5$  $T_6$  $T_7$  $T_8$  $T_9$  $T_{10}$
**Test Suite Adequacy**

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)
## Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
<th>$T_8$</th>
<th>$T_9$</th>
<th>$T_{10}$</th>
</tr>
</thead>
</table>

| $R_1$ | $R_2$ |
Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$
Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$  $T_2$  $T_3$  $T_4$  $T_5$  $T_6$  $T_7$  $T_8$  $T_9$  $T_{10}$

$R_1$  $R_2$  $R_3$  $R_4$  $R_5$  $R_6$
Test Suite Adequacy

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

\( T_1 \) \( T_2 \) \( T_3 \) \( T_4 \) \( T_5 \) \( T_6 \) \( T_7 \) \( T_8 \) \( T_9 \) \( T_{10} \)

\( R_1 \) \( R_2 \) \( R_3 \) \( R_4 \) \( R_5 \) \( R_6 \) \( R_7 \) \( R_8 \)
Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$
**Test Suite Adequacy**

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$
Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Test Suite Execution

\[ T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6 \quad T_7 \quad T_8 \quad T_9 \quad T_{10} \]
Test Suite Execution

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)
Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

JUnit Test Automation Framework
Test Suite Execution

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

JUnit Test Automation Framework

Run Test Case
Test Suite Execution

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

JUnit Test Automation Framework

Passing Test Case: \( O_A = O_E \)
Test Suite Execution

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

JUnit Test Automation Framework
Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

JUnit Test Automation Framework

Kapfhammer
Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

JUnit Test Automation Framework

Run Test Case
Test Suite Execution

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

JUnit Test Automation Framework

Failing Test Case: \( O_A \neq O_E \)
Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

Stop Running $T$
Test Suite Execution

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

JUnit Test Automation Framework

Failing Test Case: \( O_A \neq O_E \)

Stop Running \( T \)
Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

Stop Running $T$
Test Suite Execution

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

JUnit Test Automation Framework

Failing Test Case: \( O_A \neq O_E \)

Stop Running \( T \)
Continue Running \( T \)
Test Suite Execution

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

JUnit Test Automation Framework

Failing Test Case: \( O_A \neq O_E \)

Stop Running \( T \)  
Continue Running \( T \)
Test Coverage Monitoring

$T_1$  $T_2$  $T_3$  $T_4$  $T_5$  $T_6$  $T_7$  $T_8$  $T_9$  $T_{10}$
### Test Coverage Monitoring

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

| \( T_1 \) | \( T_2 \) | \( T_3 \) | \( T_4 \) | \( T_5 \) | \( T_6 \) | \( T_7 \) | \( T_8 \) | \( T_9 \) | \( T_{10} \) |
Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

JUnit Test Automation Framework
Cobertura Test Coverage Monitor
Proteja Test Suite Manager
Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

JUnit Test Automation Framework
Cobertura Test Coverage Monitor
Proteja Test Suite Manager

Run Test Case
Collect Per-Test Case Coverage
Test Coverage Monitoring

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)
Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R$ for ... Statement Coverage
Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R$ for ... Mutation Coverage
Test Coverage Monitoring

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

Requirements Set \( R \) for ... Definition-Use Coverage
Importance of Test Suite Prioritization

Prioritize to **increase** the CE of a test suite $CE = \frac{Actual}{Ideal} \in [0, 1]$
Importance of Test Suite Prioritization

Original ordering exhibits poor effectiveness score - $CE = 0.3789$
**Importance of Test Suite Prioritization**

Different ordering improves the effectiveness score - CE = 0.5053
Importance of Test Suite Prioritization

Some orderings have less improved scores - $CE = 0.4316$
Importance of Test Suite Prioritization

Test Orderings

1, 2, 3
1, 3, 2
2, 3, 1
3, 1, 2

Best ordering shows a higher effectiveness scores - CE = 0.5789
Importance of Test Suite Prioritization

Greedy methods often produce high-effectiveness orderings.
Importance of Test Suite Prioritization

Search-based techniques may have some desirable characteristics

Kapfhammer
Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Greedy Algorithms

$T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6 \quad T_7 \quad T_8 \quad T_9 \quad T_{10}$
# Greedy Algorithms

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

<table>
<thead>
<tr>
<th>( T_1 )</th>
<th>( T_2 )</th>
<th>( T_3 )</th>
<th>( T_4 )</th>
<th>( T_5 )</th>
<th>( T_6 )</th>
<th>( T_7 )</th>
<th>( T_8 )</th>
<th>( T_9 )</th>
<th>( T_{10} )</th>
</tr>
</thead>
</table>

Kapfhammer

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Greedy Algorithms

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

\( T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6 \quad T_7 \quad T_8 \quad T_9 \quad T_{10} \)

\( R_1 \quad R_2 \quad R_3 \quad R_4 \quad R_5 \quad R_6 \quad R_7 \quad R_8 \quad R_9 \quad R_{10} \quad R_{11} \quad R_{12} \)
Greedy Algorithms

**Test Suite** $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

**Requirements Set** $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$
Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Greedy Algorithms

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

Requirements Set \( R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \} \)
Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$, $T_2$, $T_3$, $T_4$, $T_5$, $T_6$, $T_7$, $T_8$, $T_9$, $T_{10}$

$R_1$, $R_2$, $R_3$, $R_4$, $R_5$, $R_6$, $R_7$, $R_8$, $R_9$, $R_{10}$, $R_{11}$, $R_{12}$

$\text{ratio}(T_1) = \frac{\text{cost}(T_1)}{\text{coverage}(T_1)}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Kapfhammer Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Key Algorithms

Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$

$ratio(T_1) = \frac{5}{3}$
### Greedy Algorithms

**Test Suite** $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

- **Requirements Set** $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$

- $\text{ratio}(T_1) = 1.66$
Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Key Algorithms

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$

1.66

$\text{ratio}(T_5) = \frac{\text{cost}(T_5)}{\text{coverage}(T_5)}$
## Greedy Algorithms

### Test Suite

\[
T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle
\]

### Requirements Set

\[
R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}
\]

\[
\text{ratio}(T_5) = \frac{8}{3}
\]

Kapfhammer Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
**Greedy Algorithms**

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$

$\text{ratio}(T_5) = 2.66$
Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
**Greedy Algorithms**

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

\[
T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6 \quad T_7 \quad T_8 \quad T_9 \quad T_{10}
\]

1.66 2.66

\( ratio(T_1) < ratio(T_5) \)

Prefer \( T_1 \) over \( T_5 \)

Requirements Set \( R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \} \)
**Greedy Algorithms**

Test Suite \( T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle \)

Requirements Set \( R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \} \)

Proceed incrementally, picking the test case with the lowest ratio value for the uncovered requirements.
Greedy Algorithms

Test Suite $T = \langle T_8, T_4, T_9, T_1, T_{10}, T_3, T_7, T_2, T_6, T_5 \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Greedy Algorithms

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$
Greedy Algorithms

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$
Greedy Algorithms

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Greedy Algorithms

Requirements Set \( R = \{R_1, R_2, \ldots, R_{11}, R_{12}\} \)
**Greedy Algorithms**

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$
Key Algorithms

Greedy Algorithms

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$
Greedy Algorithms

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Greedy Algorithms

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$
Greedy Algorithms

Key Algorithms

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$
Greedy Algorithms

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$
Greedy Algorithms

Test Suite $T = \langle T_8, T_4, T_9, T_1, T_{10}, T_3, T_7 \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Greedy Algorithms

Test Suite $T = \langle T_8, T_4, T_9, T_1, T_{10}, T_3, T_7 \rangle$

Requirements Set $R = \{ R_1, R_2, \ldots, R_{11}, R_{12} \}$
Limitations of Greedy Algorithms

Possible configuration of the coverage report
Limitations of Greedy Algorithms

Possible configuration of the **coverage report**
Limitations of Greedy Algorithms

Possible configuration of the coverage report
Limitations of Greedy Algorithms

Execution time of the test cases may mislead greedy
Limitations of Greedy Algorithms

Execution time of the test cases may mislead greedy
Limitations of Greedy Algorithms

Original ordering has low effectiveness score
Limitations of Greedy Algorithms

Original ordering has low effectiveness score

$CE(T) = 0.54$
Limitations of Greedy Algorithms

**Greedy** method constructs suite with marginal improvement.
Limitations of Greedy Algorithms

\[ \text{time}(T_1) = 1 \]
\[ \text{time}(T_2) = 1 \]
\[ \text{time}(T_3) = 2.45 \]
\[ \text{time}(T_4) = 1 \]
\[ CE(T') = 0.55 \]

**Greedy** method constructs suite with marginal improvement
Limitations of Greedy Algorithms

**Greedy** can exhibit high run-times (Jiang et al. ASE 2009)
Limitations of Greedy Algorithms

Genetic may find better orderings (Conrad et al. GECCO 2010)
Limitations of Greedy Algorithms

Genetic may find better orderings (Conrad et al. GECCO 2010)
Limitations of Greedy Algorithms

Search-based algorithms are amenable to parallelization
Limitations of Greedy Algorithms

Search-based algorithms support “human in the loop”
Limitations of Greedy Algorithms

Search-based algorithms construct diverse test orderings

Key Algorithms

CE($T'$) = 0.63
Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point
Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point

$$\langle T_1, T_2, T_3, T_4, T_5 \rangle$$
Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point

\[ \langle T_2, T_1, T_3, T_4, T_5 \rangle \]

\[ \langle T_1, T_2, T_3, T_4, T_5 \rangle \]
Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point

\[ \langle T_2, T_1, T_3, T_4, T_5 \rangle \]

\[ \langle T_1, T_2, T_3, T_4, T_5 \rangle \rightarrow \langle T_3, T_2, T_1, T_4, T_5 \rangle \]
Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point

\[ \langle T_2, T_1, T_3, T_4, T_5 \rangle \]

\[ \langle T_1, T_2, T_3, T_4, T_5 \rangle \rightarrow \langle T_3, T_2, T_1, T_4, T_5 \rangle \]

\[ \langle T_4, T_2, T_3, T_1, T_5 \rangle \]
Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point

\[ \langle T_2, T_1, T_3, T_4, T_5 \rangle \]

\[ \langle T_5, T_2, T_3, T_4, T_1 \rangle \]

\[ \langle T_1, T_2, T_3, T_4, T_5 \rangle \]

\[ \langle T_3, T_2, T_1, T_4, T_5 \rangle \]

\[ \langle T_4, T_2, T_3, T_1, T_5 \rangle \]
Hill Climbing Algorithm

\[ \text{fitness} \]

\[ \text{time} \]
Hill Climbing Algorithm

\[ \text{fitness} \]

\[ T \]
Hill Climbing Algorithm

fitness

T

T'

time

Kapfhammer

Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Hill Climbing Algorithm

\[ fitness \]

\[ T \rightarrow T' \rightarrow T'' \]

\[ time \]
Hill Climbing Algorithm

Key Algorithms

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Hill Climbing Algorithm

![Diagram of Hill Climbing Algorithm]

- Fitness (y-axis)
- Time (x-axis)

Key Algorithms

- Hill Climbing Algorithm

Kapfhammer
Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
# How Do I Evaluate Regression Testing Methods?

## Model for Experimentation

<table>
<thead>
<tr>
<th>Start</th>
</tr>
</thead>
</table>

Kapfhammer

Allegheny College

Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
How Do I Evaluate Regression Testing Methods?

Start

Programs
Test Suites
How Do I Evaluate Regression Testing Methods?

Model for Experimentation

Start

Programs
Test Suites

Conduct Experiments
How Do I Evaluate Regression Testing Methods?
How Do I Evaluate Regression Testing Methods?

Start → Programs → Conduct Experiments → Data Sets → Technique Configurations → Regression Testing Techniques

Kapfhammer
Allegheny College
Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
How Do I Evaluate Regression Testing Methods?
How Do I Evaluate Regression Testing Methods?

Model for Experimentation

Start

Programs
Test Suites

Conduct Experiments

Data Sets

Regression Testing Techniques

Technique Configurations

Visualization
Statistical Analysis

Summarized Data Sets

Graphs
Diagrams

Models
How Do I Evaluate Regression Testing Methods?

Iteratively Perform Visualization and Statistical Analysis

Start

Programs
Test Suites

Conduct Experiments

Data Sets

Regression Testing Techniques

Technique Configurations

Visualization
Statistical Analysis

Repeat

Summarized Data Sets

Graphs
Diagrams

Models
How Do I Evaluate Regression Testing Methods?

Conduct Experiments with Additional Programs, Test Suites, and Techniques
How Do I Evaluate Regression Testing Methods?

Conduct Experiments with Additional Programs, Test Suites, and Techniques

Our tools support all of these tasks!
How Do I Evaluate Regression Testing Methods?

Conduct Experiments with Additional Programs, Test Suites, and Techniques

Start

Programs
Test Suites
Conduct Experiments
Data Sets
Regression Testing Techniques
Technique Configurations
Visualization
Statistical Analysis
Summarized Data Sets
Graphs
Diagrams
Models
Repeat

End

Greedy, Hill Climbing, Random, Adaptive Random, Simulated Annealing, Genetic

Kapfhammer
Allegheny College
Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
# Case Study Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Lines</th>
<th>Methods</th>
<th>Classes</th>
<th>Faults</th>
<th>Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommissionEmployee</td>
<td>34</td>
<td>18</td>
<td>2</td>
<td>95</td>
<td>15</td>
</tr>
<tr>
<td>Point</td>
<td>125</td>
<td>26</td>
<td>3</td>
<td>44</td>
<td>13</td>
</tr>
<tr>
<td>DataStructures</td>
<td>189</td>
<td>57</td>
<td>8</td>
<td>324</td>
<td>106</td>
</tr>
<tr>
<td>Employee</td>
<td>192</td>
<td>52</td>
<td>7</td>
<td>84</td>
<td>14</td>
</tr>
<tr>
<td>LoopFinder</td>
<td>193</td>
<td>26</td>
<td>4</td>
<td>34</td>
<td>13</td>
</tr>
<tr>
<td>Sudoku</td>
<td>231</td>
<td>58</td>
<td>4</td>
<td>414</td>
<td>25</td>
</tr>
<tr>
<td>JDepend</td>
<td>1,462</td>
<td>282</td>
<td>35</td>
<td>2,659</td>
<td>39</td>
</tr>
<tr>
<td>Reduction and Prioritization</td>
<td>2,050</td>
<td>211</td>
<td>19</td>
<td>1,412</td>
<td>38</td>
</tr>
<tr>
<td>Barbecue</td>
<td>2,501</td>
<td>422</td>
<td>59</td>
<td>18,312</td>
<td>140</td>
</tr>
<tr>
<td>JodaTime</td>
<td>12,687</td>
<td>3,644</td>
<td>223</td>
<td>20,894</td>
<td>206</td>
</tr>
<tr>
<td>CommonsMath</td>
<td>20,763</td>
<td>4,185</td>
<td>556</td>
<td>6,077</td>
<td>268</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>40,427</td>
<td>8,981</td>
<td>920</td>
<td>50,349</td>
<td>877</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>3,675</td>
<td>816</td>
<td>84</td>
<td>4,577</td>
<td>80</td>
</tr>
</tbody>
</table>
Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>
## Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>
## Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>
## Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>
## Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>
# Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>
### Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>

Greedy and hill climbing produce comparable orderings
Concrete Examples

Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>

However, hill climbing is slightly more efficient than greedy
### Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>

Greedy produces a slightly better ordering than hill climbing
## Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>

But the hill climbing algorithm executes over four times faster!
## Concrete Examples

### Empirical Results: Greedy and Search-Based

<table>
<thead>
<tr>
<th>Prioritizer</th>
<th>NAPSC</th>
<th>Runtime (sec)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRD</td>
<td>0.98</td>
<td>0.22</td>
<td>Sudoku</td>
</tr>
<tr>
<td>GRD</td>
<td>0.36</td>
<td>0.38</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>GRD</td>
<td>0.71</td>
<td>33.88</td>
<td>JodaTime</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.98</td>
<td>0.01</td>
<td>Sudoku</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.36</td>
<td>0.04</td>
<td>ReductionAndPrioritization</td>
</tr>
<tr>
<td>HC_SA_FS</td>
<td>0.69</td>
<td>7.88</td>
<td>JodaTime</td>
</tr>
</tbody>
</table>

A small NAPSC increase may result in a large runtime increase
Empirical Results: Random and Adaptive Random

<table>
<thead>
<tr>
<th>Population Size</th>
<th>NAPSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.3240</td>
</tr>
<tr>
<td>20</td>
<td>0.3260</td>
</tr>
<tr>
<td>30</td>
<td>0.3265</td>
</tr>
</tbody>
</table>

Original NAPSC Score: 0.2784

Kapfhammer
Allegheny College
Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
Empirical Results: Random and Adaptive Random

Negligible NAPSC increase as population size increases
Empirical Results: Random and Adaptive Random

Increases in runtime are more marked
Empirical Results: Random and Adaptive Random

<table>
<thead>
<tr>
<th>Statement</th>
<th>JodaTime (JT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAPSC</td>
<td>0.3194</td>
</tr>
<tr>
<td></td>
<td>0.3195</td>
</tr>
<tr>
<td></td>
<td>0.3196</td>
</tr>
<tr>
<td></td>
<td>0.3197</td>
</tr>
</tbody>
</table>

Similarity Metric

Original NAPSC Score: 0.2784

Euclidean Jaccard Manhattan
Empirical Results: Random and Adaptive Random

NAPSC changes little as similarity metric is varied
Empirical Results: Random and Adaptive Random

Scores are comparable to those produced by random (0.3240 - 0.3265)
Empirical Results: Random and Adaptive Random

Adaptive random executes more slowly than random.
Empirical Results: Random and Adaptive Random

Choose random because it produces comparable NAPSC scores in less time.

Kapfhammer
Allegheny College
Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing
The Future of Regression Testing

### Research
- Reproducible research by releasing software tools and data
- Integrate many existing algorithms into a single framework
- Develop new forums for publishing important results
  - Software Quality Journal special issue
  - International Workshop on Regression Testing

### Practice
- Encourage the use of coarse-grained information
- Try to apply existing tools to industrial programs
- Participate in community events; publish experience reports
The Future of Regression Testing

Research

- Reproducible research by releasing software tools and data
- Integrate many existing algorithms into a single framework
- Develop new forums for publishing important results
  - Software Quality Journal special issue
  - International Workshop on Regression Testing

Practice

- Encourage the use of coarse-grained information
- Try to apply existing tools to industrial programs
- Participate in community events; publish experience reports
Conclusions and Future Work

Concluding Remarks

- Comprehensive framework for regression testing
- Interesting empirical results demonstrate trade-offs
- Free/open source tools are available for download
  - http://proteja.googlecode.com
  - http://modificare.googlecode.com

Future Work

- Add new algorithms for regression testing
- Conduct experiments with more case study applications
- Further develop statistically meaningful empirical results
Conclusions and Future Work

Concluding Remarks

- Comprehensive framework for regression testing
- Interesting empirical results demonstrate trade-offs
- Free/open source tools are available for download
  - http://proteja.googlecode.com
  - http://modificare.googlecode.com

Future Work

- Add new algorithms for regression testing
- Conduct experiments with more case study applications
- Further develop statistically meaningful empirical results
Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing

Gregory M. Kapfhammer

http://www.cs.allegheny.edu/~gkapfham/

Thank you for your attention!
Contact me with questions and/or comments!